

Universidad
Rey Juan Carlos

ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA
INFORMÁTICA

GRADO EN INGENIERÍA INFORMÁTICA

CURSO 2019/2020

Trabajo de Fin de Grado

“Sistema software para segmentar la pizarra y el profesor en
secuencias de vídeos de clases magistrales”

Autora:

Lara Gutiérrez Haro

Tutores:

José F. Vélez

Micael Gallego

Agradecimientos

Agradecer a José Vélez la propuesta de este proyecto ya que sin su ayuda y consejos este trabajo habría sido más tedioso. Es un placer poder contar con profesionales como él, siempre dispuestos a ayudar y con grandísimas ideas para mejorar el mundo en el que vivimos.

Agradecer también sus contribuciones a todos los desarrolladores de software libre que me han proporcionado herramientas para realizar este proyecto así como a familiares y amigos que han puesto su granito de arena aportando diferentes puntos de vista, ideas, posibilidades y apoyo.

Finalmente, quiero agradecer su paciencia a mi familia por darme la posibilidad de estudiar y evolucionar. A ellos les debo todos mis logros.

Resumen

Actualmente el uso de las nuevas tecnologías ha permitido que éstas estén en nuestra vida diaria de una manera más integral y casi todos los ámbitos de la vida cotidiana se están adaptando al uso de las mismas. Aspectos como escribir un mensaje, programar tu agenda o escuchar música ya prácticamente no se conciben sin el uso de la tecnología.

El sector educativo no se ha quedado atrás. Cada vez más personas consumen de manera habitual contenido con el que mejorar su formación a través de Internet, ya sea por medio de los MOOCs, Webinars o formación universitaria.

Este TFG tiene como objetivo mejorar la experiencia del alumnado a la hora de participar en una clase magistral de manera online. Para ello, se propone un sistema de software que permita procesar los vídeos de una clase magistral para disponer de una visión separada y mejorada tanto del profesor como de la pizarra. El objetivo de esta separación es el de lograr una mayor comodidad del alumnado durante la clase online.

1.	Introducción	5
1.2	Motivación	5
1.3	Estado del arte	6
1.4	Objetivos	7
2.	Análisis.....	9
2.1	Requisitos funcionales.....	9
2.2	Requisitos no funcionales.....	10
2.2.1	Usabilidad.....	10
2.2.2	Eficiencia.....	10
2.2.3	Seguridad.....	10
2.2.4	Robustez.....	11
2.3	Requisitos técnicos.....	12
2.3.1	Vídeos en 4K.....	12
2.3.2	Código abierto.....	13
2.3.3	Requisitos de software y hardware mínimos.....	13
2.4	Metodología	13
2.5	Soluciones deseadas	14
2.6	Diagrama de Casos de Uso.....	15
2.6.1	Alumno.....	15
2.6.2	Software de visión.....	16
2.6.3	Técnico de visión	16
3.	Diseño e implementación	18
3.1	Diagrama de Actividad.....	18
3.2	Herramientas utilizadas	20
3.2.1	Python	20
3.2.2	Pycharm.....	21
3.2.3	OpenCV.....	22
3.2.4	Numpy	22
3.3	Arquitectura del software	22
3.3.1	Diagrama de clases.....	23
3.3.2	Diagrama UML de despliegue	23
3.4	Funcionamiento del software	25
3.4.1	Consideraciones previas a la ejecución	25

3.4.2 Segmentación del profesor	27
3.5 Segmentación de la pizarra	30
4. Experimentos y pruebas	32
4.1 Pruebas realizadas	32
4.2 Resultados obtenidos.....	33
4.3 Problemas encontrados.....	35
5. Métricas.....	41
5.1 Tiempo empleado.....	41
5.2 Métricas del código	42
6. Conclusión.....	43
6.1 Trabajos futuros	44
7. Bibliografía	45
Anexo I, Configuración de la instalación.....	48
Anexo II, Ejecución de la aplicación	49

1. Introducción

Durante este capítulo se analizará cuál es el estado actual de las clases online tanto dentro como fuera de la Universidad Rey Juan Carlos, observando los posibles problemas que presenta a este formato, así como los objetivos planteados para resolver dichos problemas.

1.2 Motivación

Este TFG surge con la motivación de mejorar la experiencia del alumno a la hora de tomar una clase magistral de manera online. Sin el uso de esta tecnología, el alumno puede tener las siguientes dificultades durante una clase online:

- Dificultad de lectura al encontrarse la pizarra alejada de la pantalla del ordenador del alumno.
- Aparecen elementos innecesarios que pueden distraer al alumno como pueden ser sillas, mesas u otros objetos.
- El profesor durante la clase se podría encontrar en posiciones diferentes al moverse por lo que entorpece la vista sobre el mismo.

Sirva como ejemplo la Figura 1 grabada de una clase en la que no se utiliza el software desarrollado durante este trabajo.



Figura 1, visión desde el punto de vista del alumno "online"

En la Figura 1 se aprecia claramente los distintos problemas que un alumno puede tener al visualizar una clase mediante un vídeo tradicional. Por un lado, para ver al profesor en todo momento la cámara está en una posición alejada y esto dificulta la lectura de la pizarra. Además, la propia luz que incide sobre el aula provoca que ciertas zonas de la pizarra se encuentren expuestas a reflejos por la propia posición de la cámara.

A su vez, hay ciertos elementos en la grabación que son totalmente innecesarios y que pueden distraer al alumno fácilmente, como puede ser la superficie que brilla en la parte inferior izquierda de la pantalla o el suelo.

Por todo ello, parece interesante disponer de un sistema que permita mejorar estos aspectos de la clase online. Para ello se propone crear un vídeo separado del profesor y otro vídeo separado de la pizarra. Así, se espera que la experiencia que tenga el alumno online sea más similar a la que tendría en una clase presencial.

1.3 Estado del arte

Actualmente no existe un software comercial que dada una imagen, más o menos panorámica que contenga un profesor y una pizarra, segmente al profesor y la pizarra en dos imágenes.

Existen estudios científicos relacionados como los de Zizhao Zhang [1] que tratan sobre la importancia que tiene poder visualizar correctamente la pizarra y cómo sería posible la creación de un software que solventara estos problemas o los de Erwin M. Bakker y Thomas S. Huang [2] sobre eliminar al profesor de la pantalla. Estos estudios no disponen de una solución comercial o pública directamente utilizable.

Para hablar de la importancia y del impacto que este software tendría es preciso hablar de los MOOC's y su posible aplicación en estos.

MOOC [3] es el acrónimo en inglés de Massive Online Open Courses, lo que traducido al castellano significa "Curso Abierto y Masivo en Línea".

En sus inicios los MOOC's estaban ligados a la investigación y a la docencia ya que los primeros [4] creadores de dichos cursos eran investigadores y docentes.

A su vez Sebastian Thrun, fundador de Udacity, fue otro de los mayores contribuyentes a la expansión de esta iniciativa.

Entre las características con las que cuentan estos cursos se encuentran:

- Aprendizaje interactivo.
- Íntegramente online.
- Pueden ser gratuitos o de pago.
- Gran variedad de temas y formatos.
- Permite la interacción con profesores y alumnos de todo el mundo.

Dentro de los MOOC's podemos distinguir varios tipos [5]:

- XMOOC: similar a la educación tradicional donde el profesor es el centro de la enseñanza.
- CMOOC: el profesor proporciona un material y la comunidad construye el conocimiento en base a este, por lo que el alumno adquiere el papel protagonista en su educación.

En concreto XMOOC es el que tendría una mayor aplicación para el software desarrollado en este TFG por su carácter tradicional con el profesor como centro de la clase.

Dentro de los MOOC's se encuentran varias plataformas que siguen diferentes metodologías o tipos de MOOC.

Por un lado, se encuentra Coursera [6] como una de las mayores precursoras de estos, que siguen una metodología prácticamente uniforme en sus cursos. Estos cursos se componen de vídeos con la clase, textos con explicaciones de conceptos y tareas para practicar.

En cuanto a los vídeos, en muchos casos, se encuentran previamente grabados de una clase de universidad, por lo que la aplicación del software desarrollado en este TFG en esta plataforma podría ser viable.

Otras plataformas como MiriadaX [7] siguen una metodología similar a Coursera, pero con más contenido en español, y podría ser opción para este TFG.

1.4 Objetivos

Tras la motivación y el estado del arte expuestos anteriormente, se procede a enumerar los objetivos que con este TFG se pretenden alcanzar:

- Generar una secuencia aislada del profesor durante la clase:

El objetivo consistirá en detectar al profesor en la escena principal y después segmentar su imagen del resto de la misma.

- Generar una secuencia aislada de la pizarra durante la clase:

El objetivo consistirá en segmentar la imagen de la pizarra en la escena principal, corregir la perspectiva, y mejorar la relación del contraste.

Ambos objetivos tienen como fin último facilitar la atención del alumno ya que se eliminarán elementos innecesarios de la imagen con lo que se podrá reducir las distracciones durante el visionado de la clase.

Como muestra la Figura 2, los pasos a gran escala que sigue este proyecto para alcanzar los objetivos anteriormente mencionados son los siguientes:

Partiendo de un vídeo grabado de una clase magistral como entrada, se aplica el software desarrollado de visión y se obtienen dos vídeos de salida independientes del profesor y la pizarra.

1. Partiendo de uno vídeo grabado

2. Aplicando este software de visión creado

3. Obtener dos vídeos independientes de la pizarra y del profesor

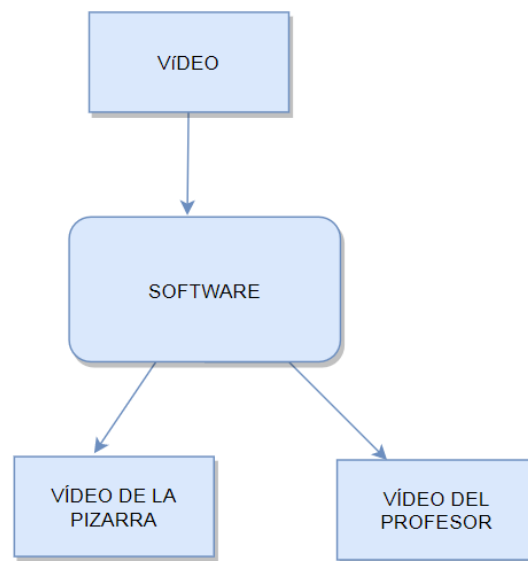


Figura 2, esquema a seguir

2. Análisis

Antes de proceder a la realización del software se debe elaborar un análisis de lo que se quiere conseguir con el fin de poder planificar y evaluar el desarrollo una vez esté finalizado.

Para ello, se establecerán una serie de requisitos, funcionales, no funcionales y técnicos. A su vez se establecerá una metodología, la solución deseada y los diferentes casos de uso.

2.1 Requisitos funcionales

Se entiende como requisitos funcionales aquellos que harán que se cumpla el comportamiento esperado del software desarrollado. A continuación se detalla cada uno de los establecidos:

Por norma general, la cámara no estará alineada con la pizarra, en el caso de que eso ocurra, se podrá corregir la desviación de la misma, con el fin de que el alumno tenga una visión de la pizarra lo más centrada posible. El primer requisito por tanto será el siguiente:

- Construir un software para segmentar y corregir la perspectiva de la pizarra.

Normalmente se dispone de una secuencia de vídeo que incluye la imagen del profesor y de la pizarra. Sería preferible tener la figura del profesor lo más aislada posible, con el fin de que el alumno pudiese centrar su atención en el profesor cuando sea necesario. Para ello se plantea el siguiente requisito:

- Construir un programa para segmentar la imagen del profesor en una secuencia de vídeo.

Las pizarras, al ser blancas y brillantes habitualmente, reflejan la luz del aula y pueden ocasionar problemas de contraste, los cuales deberán ser evitados o en su defecto ser mejorados lo máximo posible. Para solucionar este problema, se requiere:

- Construir un software para mejorar la relación de contraste de las pizarras blancas.

Cada aula contará con unas determinadas características, y el software tendrá que ser lo suficientemente flexible y robusto para afrontar estas discrepancias, en este caso por medio de un fichero de configuración. Esto da lugar al último requisito funcional:

- Poder configurar la posición de la pizarra en un archivo de configuración que cambiará para cada aula.

2.2 Requisitos no funcionales

Los requisitos no funcionales son aquellos que representarán características generales y restricciones del sistema que se esté desarrollando, por lo que no definirán en ningún caso aspectos funcionales del mismo. Se procede a explicar a continuación aquellos requisitos no funcionales que se han tenido en cuenta para el desarrollo.

2.2.1 Usabilidad

La usabilidad es un aspecto clave en la elaboración de este software. Es desarrollar una herramienta que solvante los problemas detectados y se ejecute sin grandes dificultades.

En el caso de este software el usuario final será un técnico, el cual deberá manejar sin grandes dificultades el sistema. A pesar de requerir unos ajustes necesarios a través de unos ficheros de configuración, no suponen ningún reto en cuanto a usabilidad para una persona familiarizada con aspectos técnicos.

2.2.2 Eficiencia

El programa debe ser capaz de procesar y ofrecer el resultado esperado en unos márgenes de tiempo aceptables. Es decir, el programa debe procesar un vídeo en un tiempo razonable, a ser posible cercano al real. En el capítulo cuarto se detallan las pruebas que se han realizado a este respecto.

2.2.3 Seguridad

Desde el punto de vista de la seguridad, a nivel de software no se ha requerido un desarrollo extra, ya que la herramienta será de uso interno y será ejecutada en local sin ninguna conectividad, por lo que no hay riesgo de filtración de datos de ningún tipo.

2.2.4 Robustez

Uno de los problemas que se presentan en los vídeos analizados son los diferentes cambios de iluminación que se pueden presentar durante la grabación del vídeo o entre distintos vídeos, por ello el programa deberá ser lo suficientemente robusto como para soportar tal complejidad y permitir un resultado en las distintas condiciones de luminosidad.

Como muestran la Figura 3 y la Figura 4, correspondientes a dos minutos diferentes del mismo vídeo, la iluminación a lo largo de una clase sufre variaciones. Se puede apreciar que la iluminación inicial del proyector (ver Figura 3) tiene un color azul más brillante que el que muestra posteriormente, (ver Figura 4) que posee un color negro mucho más apagado. Es por ello que este software debe ser capaz de, sufriendo cambios de iluminación el vídeo, tener siempre la imagen de la pizarra y el profesor perfectamente identificadas para procesarlas.

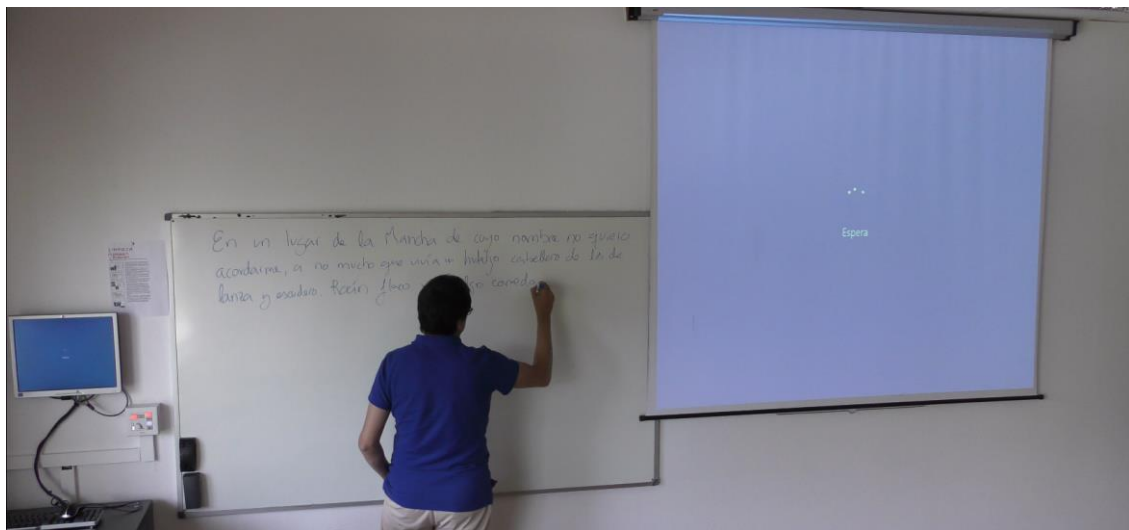


Figura 3, iluminación inicial

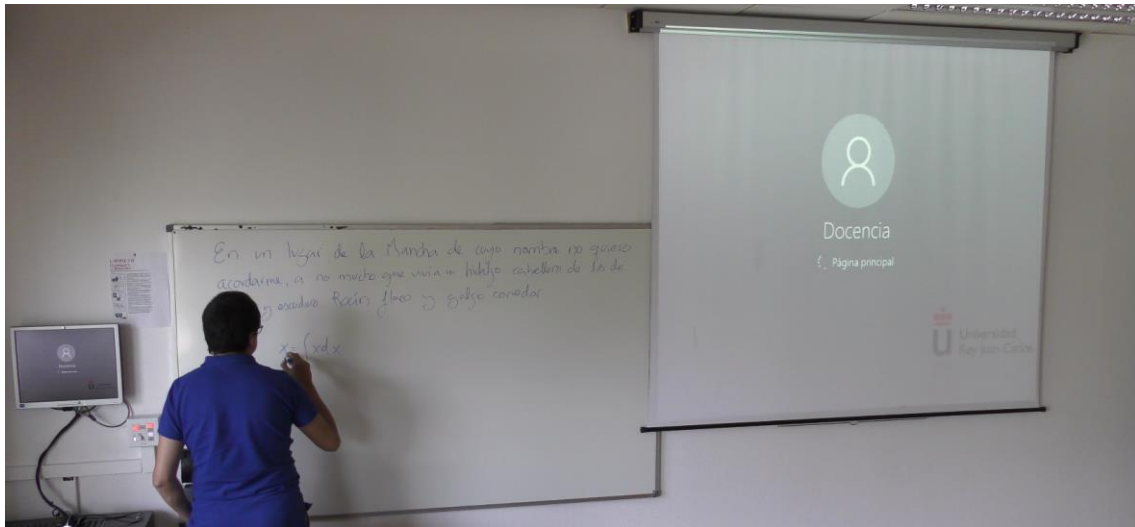


Figura 4, iluminación tras unos minutos de vídeo

2.3 Requisitos técnicos

Los requisitos técnicos serán aquellos que nos permitirán cumplir los requisitos funcionales descritos con anterioridad. Se pueden destacar los siguientes:

2.3.1 Vídeos en 4K

Los vídeos en resolución 4K [8] son aquellos que tienen un tamaño de imagen aproximado de 4.000 píxeles de resolución horizontal.

Durante el desarrollo del software se han probado dos resoluciones de imagen: 1280x720 y 3840x2160 (4k). Tan solo en 4k se ha obtenido la calidad suficiente para apreciar detalles en la pizarra con suficiente nitidez. Por ello, las clases de este TFG serán grabadas en versión 4k y el software debe ser capaz de procesarlas en tiempo razonable con esa resolución.

2.3.2 Código abierto

Se ha querido utilizar librerías de código abierto en la realización de este TFG ya que proporciona ciertas ventajas con respecto al código privativo:

- Extensa comunidad: al estar abierto hace que la barrera de entrada para aquel que quiera contribuir sea mínima, por lo que genera una mayor participación de la comunidad. Esto hace que ante cualquier duda o problema haya mayor información al respecto.
- Confianza: al haber una mayor comunidad, también hay mayor confianza en él ya que es mayor el número de personas que han examinado el código.
- Gratuito: al estar el código en abierto abarata los futuros costes que pudieran existir.

2.3.3 Requisitos de software y hardware mínimos

Era necesario que tanto a nivel de software como de hardware no hicieran falta unos requisitos altamente complejos.

A nivel de software, el técnico tan solo necesitará tener acceso con un protocolo seguro al servidor de la universidad donde se aloje tanto el software como los programas, datos y herramientas necesarias para su ejecución. Este servidor contará con PyCharm [9], Python y OpenCV instalados para que el técnico ajuste los nuevos vídeos y compruebe que son correctos antes de ser puesto a disposición del alumnado.

El alumnado necesitará acceso a internet y este podrá ser desde cualquier dispositivo que permita ver vídeos en línea. Será el servidor de la universidad encargado de almacenar estos vídeos el que proporcione la seguridad adecuada para que el acceso a ellos no suponga un problema.

2.4 Metodología

Todo este TFG ha sido elaborado desde una perspectiva y filosofía Agile. Se suele categorizar en muchos casos Agile como una metodología cuando en verdad es una filosofía [10] en sí misma, que supone una forma diferente de trabajar y de organizarse.

El principal motivo para llevar a cabo una filosofía Agile es la manera de subdividir hitos o tareas a la hora de desarrollar el programa.

Para la elaboración de este TFG es bastante lógico e intuitivo organizar en pequeñas partes el programa principal, ya que aunque el fin último es conseguir segmentar tanto al profesor como a la pizarra, cualquiera de las dos maneras individualmente estaría aportando un valor significativo de la misma manera con el resto de requisitos.

A continuación explican cuáles son las principales ventajas [11] por las que se ha llevado a cabo esta filosofía.

- Permitir dimensionar los proyectos más fácilmente:

Al tener subdividido las tareas a realizar es más sencillo proceder a establecer diferentes hitos, por lo que es más sencillo a su vez establecer plazos de presentación del mismo.

- Facilita establecer prioridades:

Relacionado con lo anterior, es más sencillo establecer prioridades entre los diferentes hitos encontrados en función de la complejidad de los mismos si estos hitos se encuentran subdivididos.

- Entregas rápidas y más visibilidad para el cliente final.
- Feedback constante y más ágil.
- Incremento de la motivación personal al ver resultados con más frecuencia.
- Reduce complejidad de cada entrega y por tanto menor probabilidad de errores.

A lo largo de este desarrollo la iteración y subdivisión de tareas ha sido una opción muy positiva ya que ha permitido trabajar previamente en cada caso que se necesitaba encontrando soluciones pequeñas que motivaban la elaboración del mismo.

El orden con el que se ha llevado a cabo la subdivisión contando cada una con diversas iteraciones ha sido el que sigue:

- Desarrollo la parte de la pizarra en distintas iteraciones
- Realización de la parte del profesor con diversas iteraciones
- Escritura de la memoria
- Creación de la presentación

2.5 Soluciones deseadas

Las soluciones deseadas se podrían subdividir brevemente en las siguientes fases:

- El software recibirá un vídeo de una clase. Este vídeo será grabado por el profesor en una clase magistral desde un dispositivo de la universidad con capacidad para ello.
- Aislamiento del profesor en la imagen.
- Aislamiento de la pizarra en la imagen.
- Mejora de la calidad de la pizarra.
- Devolverá 2 vídeos: profesor y pizarra.

2.6 Diagrama de Casos de Uso

A continuación se detallan cada uno de los actores involucrados quedando ilustrados mediante un diagrama de casos de uso que se mostrarán las funciones que tendrá el sistema a través de los actores involucrados.

2.6.1 Alumno

En el caso del alumno, como se puede ver en la Figura 5, la única acción que llevará a cabo será la visualización de los vídeos generados tras el procesamiento inicial.

El alumno accederá a una plataforma en la que previamente se han subido los vídeos tratados por un técnico y podrá tener una clase online.

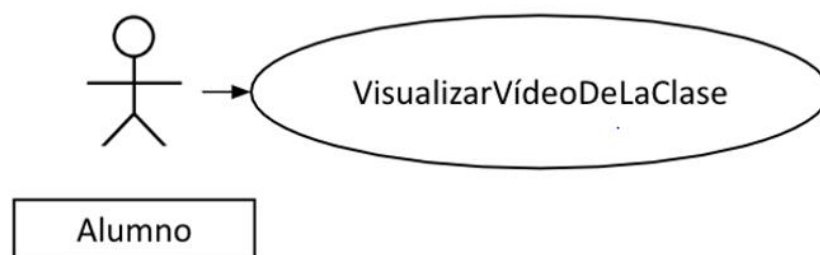


Figura 5, casos de uso de Alumno

2.6.2 Software de visión

El actor principal será el propio software ya que se encargará de segmentar la pizarra, corregir la perspectiva y mejorar el contraste de la misma, así como de segmentar al profesor. Esto se puede apreciar en la Figura 6.

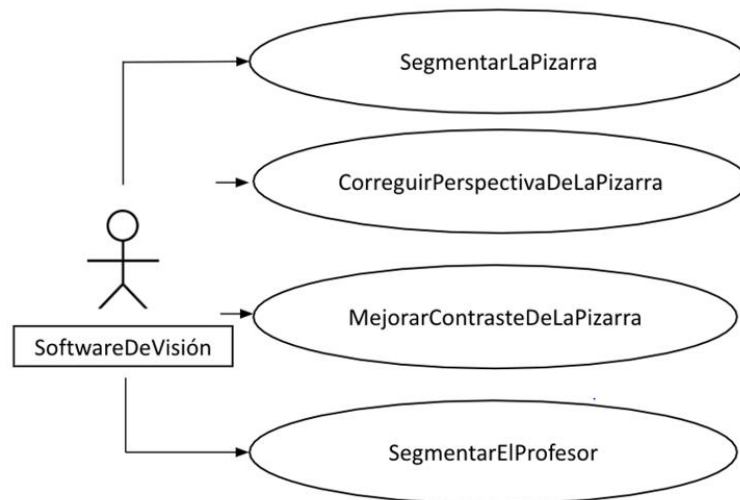


Figura 6, casos de uso del software de visión

2.6.3 Técnico de visión

El técnico de visión tendrá que configurar cada vídeo para su correcta visualización. Esta configuración consiste en el ajuste de las nuevas coordenadas de la pizarra en caso de un cambio de posición de la cámara. Gracias a su función de ajuste, el alumno podrá disponer de las clases online tratadas y almacenadas en un servidor de la universidad. Se puede percibir en la Figura 7.

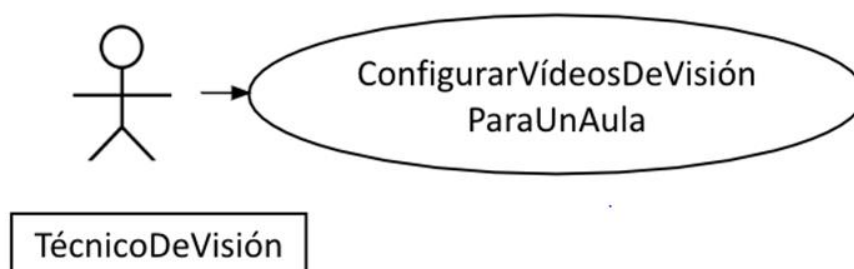


Figura 7, casos de uso del técnico de visión

Para tener una visión más alta del software, el diagrama de casos de uso unificado sería el que sigue en la Figura 8:

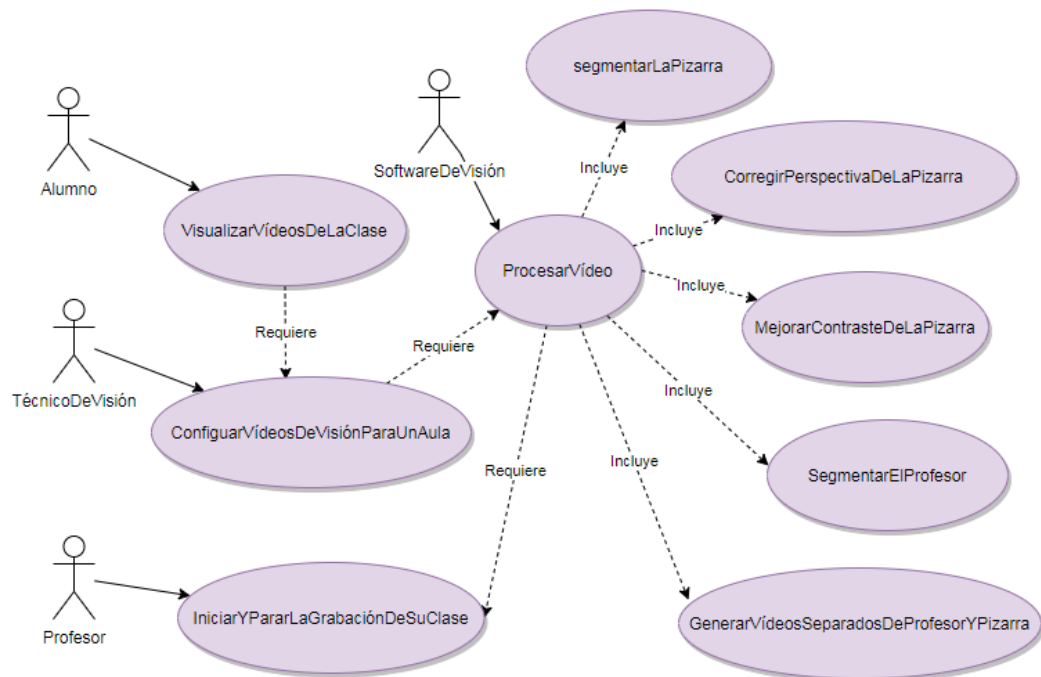


Figura 8, Diagrama de casos de uso unificado

Como se puede distinguir, el software de visión es el encargado de realizar todas las tareas necesarias para la obtención de los vídeos finales. Es necesario que previamente el profesor realice el grabado de las clases y que del técnico de visión configure estos vídeos para que el alumno, destinatario final, pueda tener acceso a ellos.

3. Diseño e implementación

Analizado el problema que se intenta resolver y establecido una serie de requisitos que tendrá el software que se desea desarrollar, en este capítulo se procede a presentar un diseño que cumpla dichos requerimientos.

Además, se detallarán las herramientas que se han utilizado para realizar el trabajo y se explicará la arquitectura del software desarrollado.

3.1 Diagrama de Actividad

A continuación se muestran los diagramas de actividad tanto a nivel general como a nivel específico para la segmentación del profesor y la pizarra.

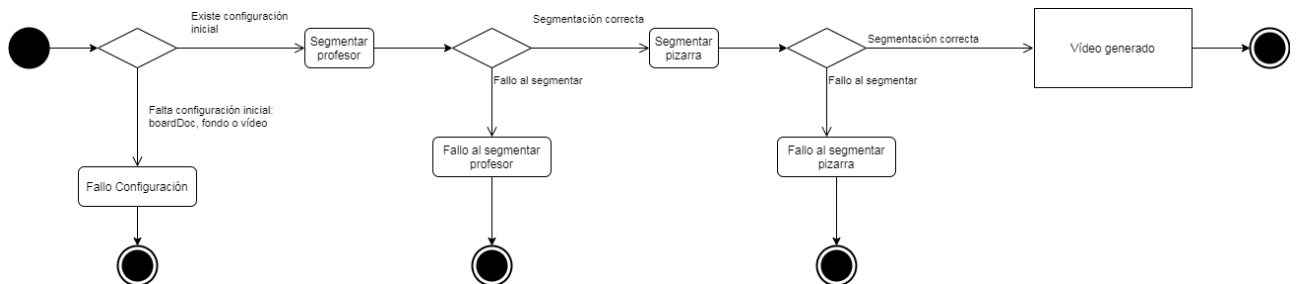


Figura 9, Diagrama de actividad general

Como se puede apreciar en la Figura 9, para la obtención de los vídeos finales tanto del profesor como de la pizarra, es necesario que el software ejecute todos los pasos sin fallo ya que este pararía el proceso y los vídeos no se generarían.

Una vez iniciado el proceso, es preciso tener un archivo de configuración XML para la pizarra y una imagen de fondo de pantalla para segmentar el profesor, si alguna de estas dos faltara, se produciría un fallo en la configuración.

Tras este primer paso, se procede a segmentar al profesor, en caso de que alguno de los métodos usados fallara, el proceso se detendría.

Si todo ha ido bien, se avanza a la segmentación de la pizarra. Este paso dará fallo si existe algún problema en el proceso. Por último, se obtendrían los dos vídeos correspondientes a la pizarra y el profesor segmentados y listos para su posterior uso.

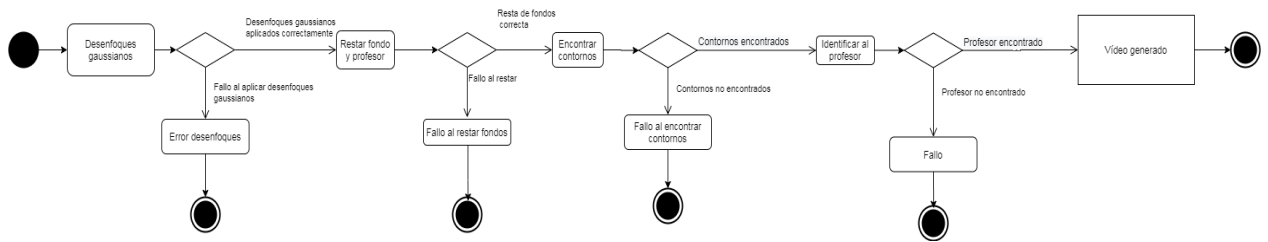


Figura 10, Diagrama de actividad de procesamiento del vídeo del profesor

Detallando el diagrama de actividad de procesamiento del vídeo del profesor, como se puede ver en la Figura 10, los pasos son los que siguen:

Se procesa la imagen del fondo de pantalla aplicando un filtro gaussiano y pasando la imagen a gris, se resta esta imagen al frame actual del vídeo, al que se le ha aplicado de la misma manera el filtro gaussiano y pasado a gris anteriormente. La imagen resultante se umbraliza y se usa para encontrar los contornos, descartar los falsos y obtener el que contiene al profesor.

Si alguno de estos pasos fallara, el software detectaría un fallo y terminaría su ejecución.

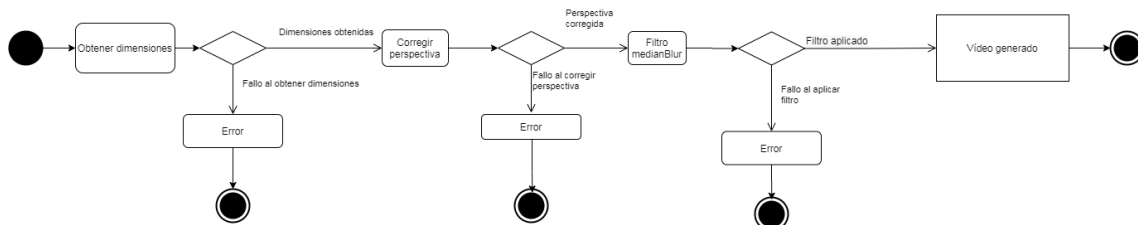


Figura 11, Diagrama de actividad del procesamiento del vídeo de la pizarra

En cuanto al diagrama de actividad del procesamiento del vídeo de la pizarra, como se muestra en la Figura 11, se ha seguido la siguiente secuencia:

Se obtienen las dimensiones de la pizarra gracias al archivo de configuración, se corrige la perspectiva, ya que la pizarra en el vídeo corresponde a un polígono mientras que en la realidad es un rectángulo, se aplica un filtro para la calidad de la misma y se obtiene un vídeo final que contiene únicamente la pizarra segmentada.

Como en el caso anterior, si alguno de los pasos para la obtención de la pizarra fallara, el software daría un error y no seguiría su ejecución.

3.2 Herramientas utilizadas

En esta sección se procede a explicar qué tecnologías, incluyendo IDE, lenguajes y paquetes, han sido utilizadas durante el desarrollo de este TFG.

3.2.1 Python

Python [12] es un lenguaje de programación interpretado, es decir, el código es convertido a lenguaje máquina a medida que es ejecutado. Al contrario que los lenguajes de programación compilados, que son convertidos a lenguaje máquina previamente, Python requiere de un paso extra antes de ser ejecutado [13].

Python se ha convertido en un lenguaje muy popular [14], y todo es gracias a las siguientes ventajas:

- Legibilidad de código:

La principal filosofía [15] por la que apuesta Python es por la legibilidad del código, lográndolo a través de una sintaxis sencilla que facilita su implementación.

- Gran versatilidad:

En comparación con otros lenguajes, Python destaca en su enorme versatilidad, pudiendo ser adaptado a varios escenarios. Algunos de los usos donde se puede encontrar son los siguientes:

- ❖ Programación web: gracias a Frameworks como Django o Flask.
- ❖ Scripting: más comúnmente usado para realizar scrapping de webs o para gestionar infraestructuras.
- ❖ Inteligencia Artificial: a pesar de tener los usos vistos anteriormente, si hay uno en el que Python destaca profundamente es en el campo de la inteligencia artificial gracias a la popularidad de técnicas como Machine Learning o Deep Learning que utilizan de forma intensiva de este lenguaje.

- Gran comunidad:

Python goza de una comunidad muy amplia y activa. Además, Python cuenta con una licencia de código abierto, lo que hace que su distribución sea completamente gratuita, incluyendo software con el que se puede comercializar. Por ello, la comunidad se encuentra involucrada en todo el proceso de mejora de Python.

- Gran cantidad de librerías:

Unido a lo anterior, gracias a la extensa comunidad con la que cuenta Python, el lenguaje se va actualizando más rápidamente y mejor y mayor es su calidad.

- Facilidad de uso:

La legibilidad y la gran cantidad de librerías existente hacen de Python un lenguaje recomendado para que aquellos que quieran iniciarse en programación, lo hagan de forma más sencilla y contando con mayor apoyo.

Gracias a las características previamente comentadas, Python está alcanzando unos índices de popularidad [16] muy altos, tan solo situándose por detrás de Java y C.

3.2.2 Pycharm

Pycharm es una IDE usado principalmente para el desarrollo de aplicaciones en Python, aunque también soporta Javascript o CoffeeScript.

Está desarrollado por JetBrains y tiene las siguientes características en su versión gratuita “PyCharm Community Edition”:

- Editor de Python inteligente.
- Depurador gráfico y ejecutor de pruebas.
- Navegación y refactorización.
- Compatibilidad con VCS.
- Soporte multiplataforma en Windows, Linux y macOS.

3.2.3 OpenCV

OpenCV [17] es una librería de Open Source para llevar a cabo tareas de Visión Artificial y Machine Learning. Desde sus inicios en el 1999 se ha utilizado en multitud de aplicaciones, entre las que destacan:

- Sistemas de seguridad con detección de movimiento.
- Control de procesos detectando objetos.
- Sistemas de seguimiento.

Al ser software libre, ha permitido que multitud de empresas hayan apostado por esta librería ya que podrían llevar a cabo modificaciones de la misma y posteriormente proceder a su comercialización sin ningún problema.

Actualmente, OpenCV contiene más de 2500 algoritmos, entre los que se encuentran:

- Detector de caras.
- Identificador de objetos.
- Extracción de modelos 3D de objetos.
- Encontrar imágenes similares para producir una final de mayor calidad.
- Seguir movimientos de los ojos.

Tras el análisis anterior tanto de Python como de OpenCV, queda demostrado el porqué del empleo de estas dos herramientas tan potentes para la realización de este TFG.

3.2.4 Numpy

Numpy [18] es una biblioteca que añade funciones matemáticas a Python. Cuenta con un área de alto nivel que permite operaciones sobre matrices y vectores. Es por ello que en data science es una de la biblioteca más utilizadas [19].

3.3 Arquitectura del software

A continuación se procede a mostrar la arquitectura del software con la ayuda de un diagrama de clases y un diagrama de despliegue.

3.3.1 Diagrama de clases

El diagrama de clases nos permite tener una visión global de las clases involucradas en el software desarrollado así como los métodos y propiedades de cada clase.

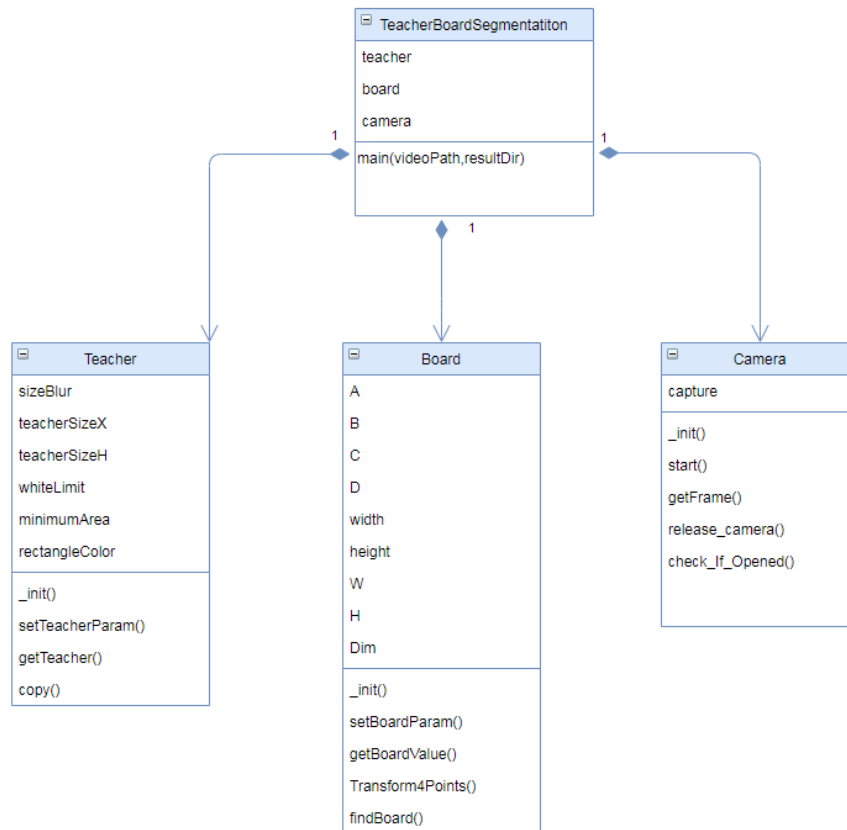


Figura 12, diagrama de clases del software implementado

Como se puede observar en la Figura 12, este software cuenta con cuatro clases:

- Clase **TeacherBoardSegmentation**. Dispone de un objeto de la clase **Teacher**, otro de la clase **Board** y otro de la clase **Camera**. Es la responsable de inicializar el proceso, llamar a las funciones y devolver los vídeos finales.
- Clase **Teacher**. Consta de una serie de variables y métodos que permiten el procesamiento de cada frame y la obtención del profesor en cada uno de ellos.
- clase **Board**. Encargada de devolver, mediante una serie de métodos señalados en el diagrama, la imagen final de la pizarra en cada frame que recibe.
- Clase **Camera**. Responsable de abrir el vídeo, comprobar que ha sido abierto correctamente y una vez acabado el proceso, hacer un release del mismo.

3.3.2 Diagrama UML de despliegue

El diagrama UML de despliegue modela la arquitectura en tiempo de ejecución del sistema implementado.

Como se puede apreciar en la Figura 13, este diagrama cuenta con la parte hardware o nodos y la parte software o artefactos que se encuentran en cada uno de los nodos.

Cada nodo debe de contener los siguientes artefactos:

- Ordenador del aula: este ordenador es propiedad de la universidad y necesitará tener un sistema para grabar vídeos. Se comunicará con el servidor de vídeos para alumnos mediante una red interna (LAN).
- Servidor de vídeos para alumnos: este servidor contará con un sistema de almacenamiento de vídeos y un sistema de protección de datos, vídeos y programas. Además, albergará el software implementado llamado TeacherBoardSegmentation.py junto con los archivos CAMARA.XML y Fondo.png necesarios para la ejecución del mismo.
- Ordenador del técnico: este ordenador debe contar con un protocolo de seguridad SSH para acceder al servidor de vídeos y poder ejecutar.
- Ordenador del alumno: este dispositivo debe usar un protocolo http para el acceso a los vídeos.

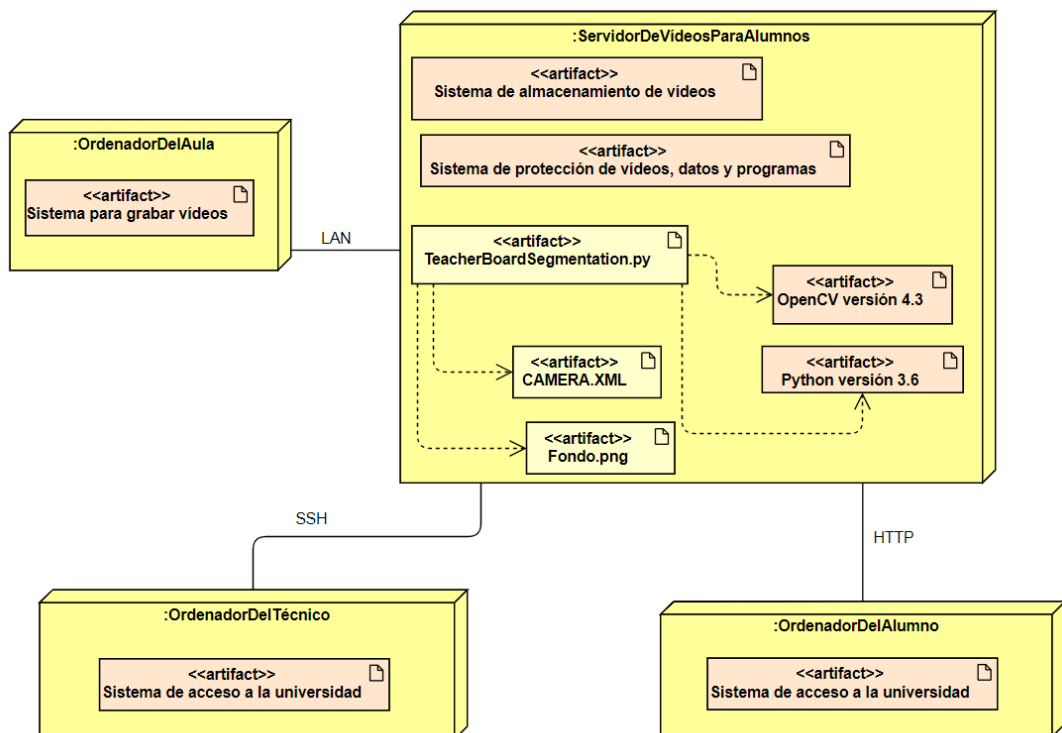


Figura 13, Diagrama UML de despliegue del software implementado

3.4 Funcionamiento del software

A lo largo de esta sección se detalla el proceso llevado a cabo desde la configuración inicial necesaria hasta la obtención de la salida esperada con el vídeo del profesor segmentado y el vídeo de la pizarra también segmentada.

3.4.1 Consideraciones previas a la ejecución

De cara a ejecutar el programa y que este genera una imagen aislada del profesor y de la pizarra, hay que tomar unas consideraciones previas que se controlarán a través de un fichero de configuración y del propio código.

XML de configuración

A la hora de procesar cada vídeo se tendrá un fichero de configuración único en función de la situación de la pizarra en el mismo ya que cada clase puede tener una distribución única con diferentes ángulos y posiciones.

El programa contendrá una carpeta llamada BoardDoc, en la que tendrá los archivos de CAMARA.XML y CAMARA2.XML correspondientes a la configuración de la pizarra para cada uno de los vídeos procesados. Estos archivos contendrán las coordenadas actuales y las coordenadas que debería tener en la realidad.

A continuación se muestran ambos ejemplos en la Figura 14 y Figura 15:

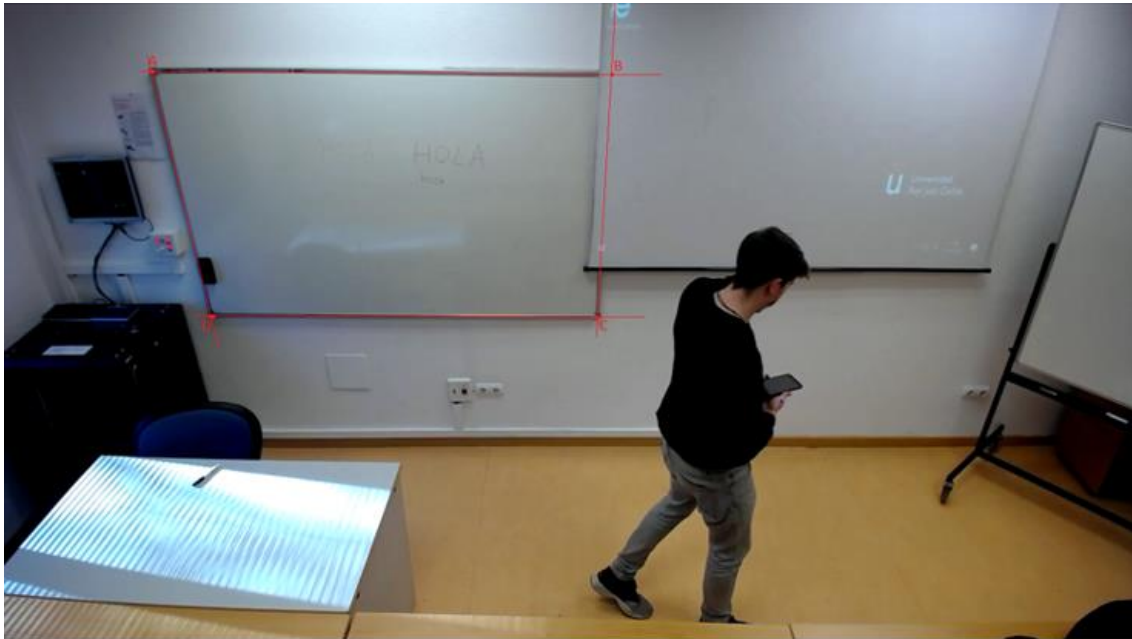


Figura 14, escena inicial 1

Como se puede comprobar en la Figura 14, la pizarra, debido a la posición de la cámara que graba, aparece como un polígono siendo en realidad un rectángulo.



Figura 15, escena inicial 2

En la Figura 15, al igual que en el caso anterior de la Figura 14, la pizarra aparece como un polígono.

En este caso, las coordenadas de la pizarra son diferentes al anterior, por este motivo será necesario que el personal técnico de visión posteriormente ajuste estos parámetros.

3.4.2 Segmentación del profesor

El primer paso será tomar una imagen de fondo (imgBack) como referencia en la cual no aparezca el profesor, para poder encontrar las diferencias posteriormente. Esta imagen vendrá de un frame tomado del propio vídeo antes y estará guardada en el proyecto como Fondo.png.

A ella se le aplica un desenfoque gaussiano [20], convolucionándola, con el cual se conseguirá una imagen con un suavizado mayor, perdiendo algunos detalles y volviéndose menos nítida. Este efecto se consigue mezclando los colores de los píxeles colindantes y es similar al efecto que provocaría una cámara desenfocada.

A continuación se procede a tratar la imagen con el profesor, como en el caso del fondo, se pasa a gris la imagen y se aplica desenfoque gaussiano, ver Figura 16.



Figura 16, escena transformada a gris

Con la imagen de fondo y la imagen con el profesor, se procede a restar ambas imágenes con el fin de comprobar si está o no el profesor. Para ello se aplicará la función absdiff de OpenCV.



Figura 17, imagen del resultado de aplicar absdiff

Tras ello, se procede a umbralizar la imagen obtenida a través del método Threshold. Este recibe una imagen, establece un límite y convierte la imagen a negro en las partes que se encuentren debajo del umbral establecido y a blanco las que se encuentren por encima (ver Figura 18). Todo este proceso es muy sensible, ya que varía enormemente según la luz que incida en la imagen y se ajusta en cada caso en particular para posteriormente poder sacar contornos.

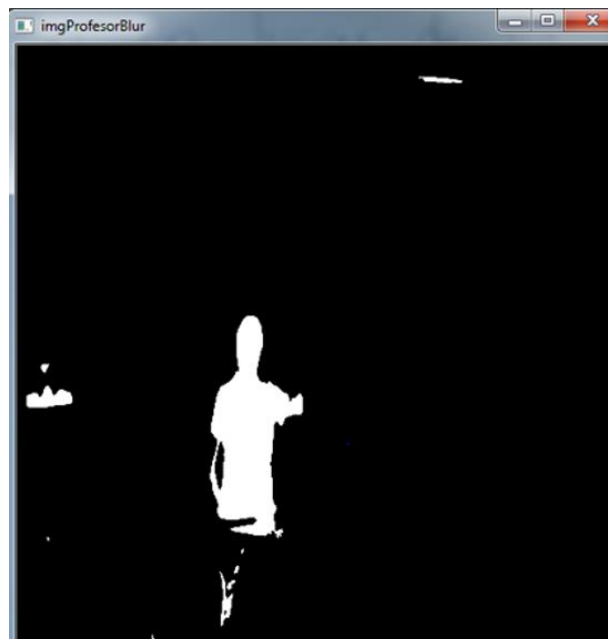


Figura 18, imagen resultado de aplicar threshold

Como paso siguiente, se buscan los contornos de la imagen anterior con la función FindContours y el método CHAIN_APPROX_SIMPLE que no obtendrá todos los puntos en la imagen, si no todos aquellos relevantes para obtener el contorno. El ejemplo que se suele utilizar para explicar dicho método es el de una línea recta, ya que para obtener los contornos de una línea no sería necesario tener todos y cada uno de los puntos que forman esa línea, si no tan solo los puntos de inicio y fin de la línea.

Una vez que se tienen todos los contornos se identifica al profesor, para lo cual se recorrerán todos los contornos encontrados y se extrae aquel contorno que tenga un área mínima, lo cual nos determine que es el profesor.

Como filtro extra, también se descartarán los contornos que se encuentren en posiciones muy distintas al anterior ya que no es posible que el profesor de un salto de un lado al otro de la cámara tanto en el eje horizontal como en el vertical ya que como ser humano, no tiene la capacidad de dar un salto tan grande.

Finalmente, con el contorno identificado, se recorta el área en el que se encuentra ese contorno y se redimensiona para que tenga un tamaño adecuado en el vídeo final (ver Figura 19).



Figura 19, profesor aislado

3.5 Segmentación de la pizarra

Una vez se tenga generado el fichero XML con la configuración de la posición de la pizarra, el primer paso será extraer dicha información. Para ello se tendrá una función que lea el archivo XML llamada `setBoardParam` y además se obtendrán las dimensiones de la pizarra, tanto las que tenga en el vídeo como las reales que debería de tener.

Como los valores de las coordenadas de la pizarra no son del todo exactos, ya que la perspectiva de la cámara no es perpendicular a la pizarra, se corregirán con la función `Transform4Points` y los puntos obtenidos del archivo de configuración.

Esta función recibe una imagen (`img`) y primeramente calcula 2 perspectivas. La primera, `pst1`, en función de lo obtenido anteriormente proveniente del fichero XML y la segunda, `pts2`, que será la variable destino una vez transformada. Para llevar a cabo la transformación se utiliza la función `getPerspectiveTransform` [21] y `wrapPerspective` de la librería OpenCV.

El último paso sería aplicar un filtro, median en este caso, que recorrerá cada pixel de la imagen y reemplazará este valor por la media entre el mismo y los vecinos colindantes. Para realizar este filtro se utiliza la función `medianBlur` de OpenCV (ver Figura 20).

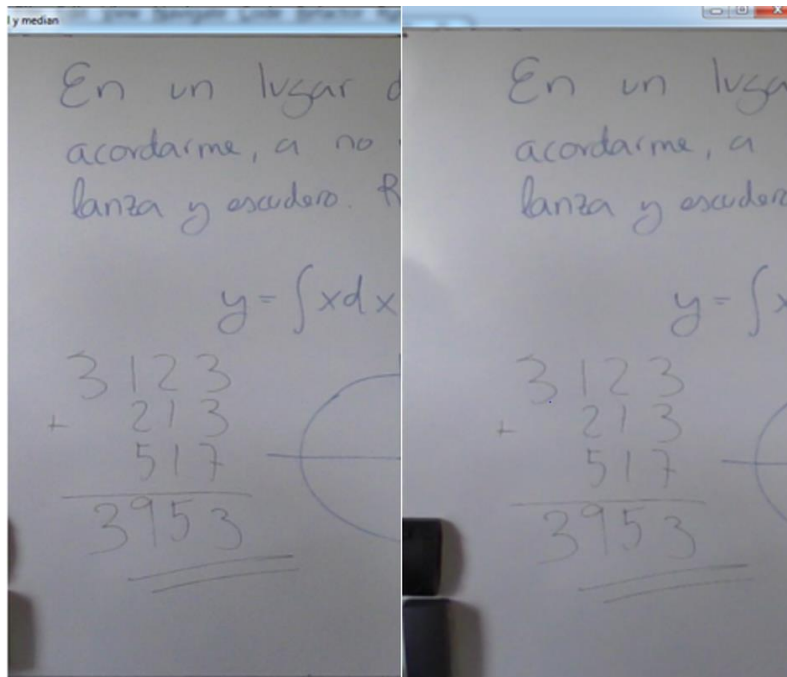


Figura 20, aplicando filtros a la pizarra

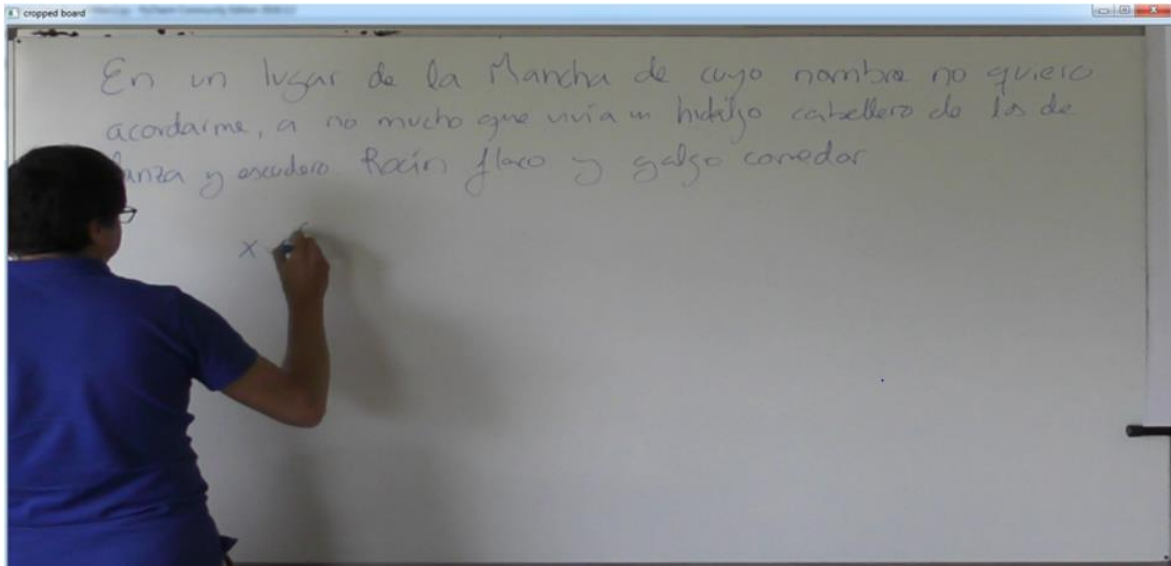


Figura 21, resultado final de la pizarra aislada

Como muestra la Figura 21, tras finalizar con la ejecución del vídeo, se obtendría la pizarra segmentada. La pizarra es perfectamente legible haciendo más legible y cómodo el seguimiento de una clase para el alumno.

4. Experimentos y pruebas

A continuación se comentan las pruebas realizadas con dos vídeos de diferentes calidades, los resultados de estos experimentos con detalle y los problemas encontrados. Tanto las pruebas como el código se encuentran en la web GitHub y es posible su descarga desde el siguiente link: <https://github.com/laraGutierrez/>

4.1 Pruebas realizadas

Se han realizado dos pruebas con dos vídeos de aulas diferentes y resolución de imagen distinta.

La prueba uno corresponde a un vídeo con resolución 1280x720 y está grabada a un aula pequeña mientras que la prueba dos se ha realizado con una resolución 4k en un aula más grande.

Para la prueba uno se ha utilizado un vídeo de 19 segundos con 454 frames y las conclusiones obtenidas son:

- 450 son frames positivos correspondientes al profesor, lo cual denota la gran capacidad del software para detectar su posición.
- 4 son frames verdaderos negativos en los que no está el profesor. Estos frames corresponden a las primeras iteraciones del software en las que se busca el contorno correspondiente.
- La pizarra está localizada en todos los frames. Gracias al archivo de configuración creado, la detección de la pizarra es precisa en todo momento.
- El tiempo de ejecución se corresponde con el del vídeo. Al tener una resolución de 1280x720, el tiempo de ejecución corresponde prácticamente al tiempo real del vídeo.

En cuanto a la prueba dos, se ha procesado un vídeo de 60 segundos en 4k con 1525 frames. Las conclusiones obtenidas son:

- 1520 son frames verdaderos positivos correspondientes al profesor. Esto quiere decir que la precisión de este software es bastante grande.
- 5 son frames falsos negativos. Para encontrar al profesor por primera vez el software debe ajustarse y estos frames corresponden a esos primeros segundos del vídeo en los que su detección no ha sido satisfactoria.
- La pizarra está localizada en todos los frames. De nuevo, como pasaba en la prueba uno, disponemos de un archivo de configuración con el cual tenemos identificada la pizarra en todo momento.
- La ejecución tarda 4 minutos. Debido a la alta resolución de este vídeo en 4K, su tiempo de ejecución se ve más afectado que en el caso anterior.

4.2 Resultados obtenidos

Tras aplicar todos los procesos descritos con anterioridad, los resultados obtenidos son los siguientes:

- Resultado final de los vídeos con resolución 1280 x 720:

Como se puede apreciar en la Figura 22 y Figura 23, tanto la pizarra como el profesor han sido segmentados del vídeo original satisfactoriamente dando como resultado dos vídeos de salida esperados.

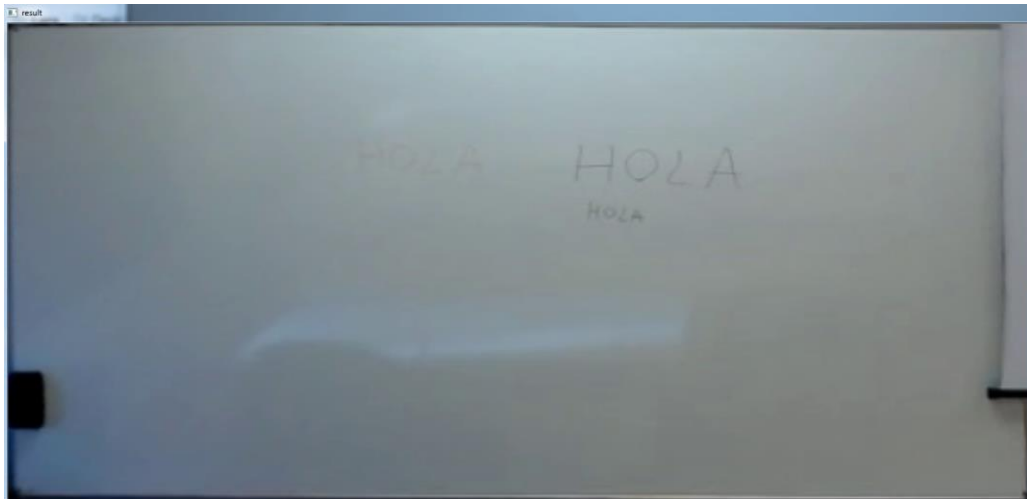


Figura 22, imagen final de la pizarra segmentada



Figura 23, resultado final de la segmentación del profesor

- Resultado final del vídeo 4k:

Como muestra la Figura 24 y Figura 25, el resultado final obtenido del vídeo con resolución 4k ha sido el esperado al segmentar tanto la pizarra como el profesor y generar ambos vídeos de salida.

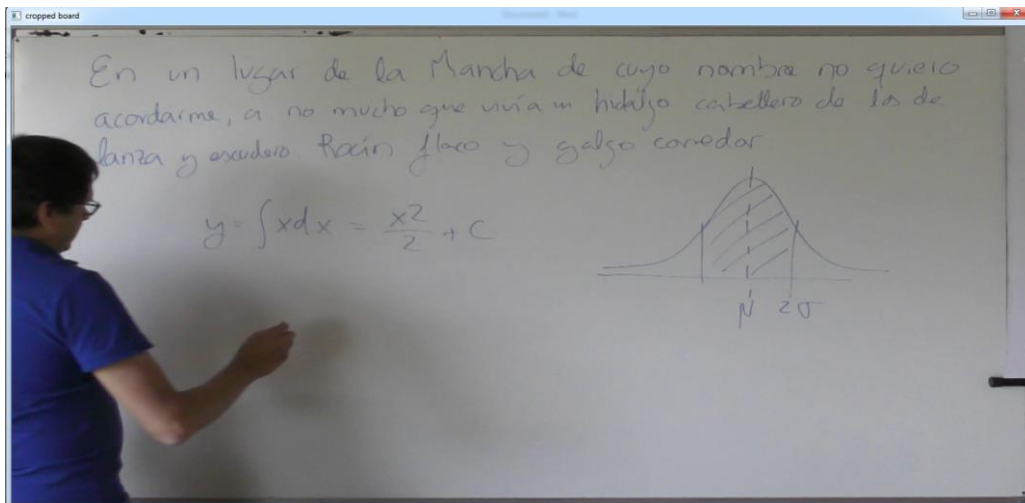


Figura 24, resultado final de la pizarra segmentada



Figura 25, resultado final del profesor segmentado

La Figura 24 y Figura 25 muestran el resultado final de los dos vídeos que se obtendrían al finalizar la ejecución del software. Como se aprecia, es posible tener al profesor identificado siempre en el mismo lugar y la vista de la pizarra constante haciendo las clases online del alumno mucho más cómodas y adecuadas.

4.3 Problemas encontrados

En esta sección se comentarán aquellos problemas encontrados a lo largo del desarrollo del software implementado.

- Aplicar el desenfoque gaussiano al profesor y falsos positivos en la segmentación del profesor:

Sin duda uno de los mayores retos es fijar el tamaño del sizeblur, que hace referencia al tamaño del Kernel en cuanto a altura y anchura. Estos valores deben ser positivos e impares y dependiendo de ellos, el difuminado de la imagen queda de una forma u otra.

Para ilustrar este problema (ver Figura 26 y Figura 27) se han unido la imagen de la pizarra segmentada (parte izquierda en ambas imágenes) y la imagen del profesor segmentada (parte derecha en ambas imágenes). En estas figuras podemos ver que cambiar el valor del Kernel generaría una pérdida de la imagen del profesor final dando como resultado un falso positivo o un positivo verdadero:

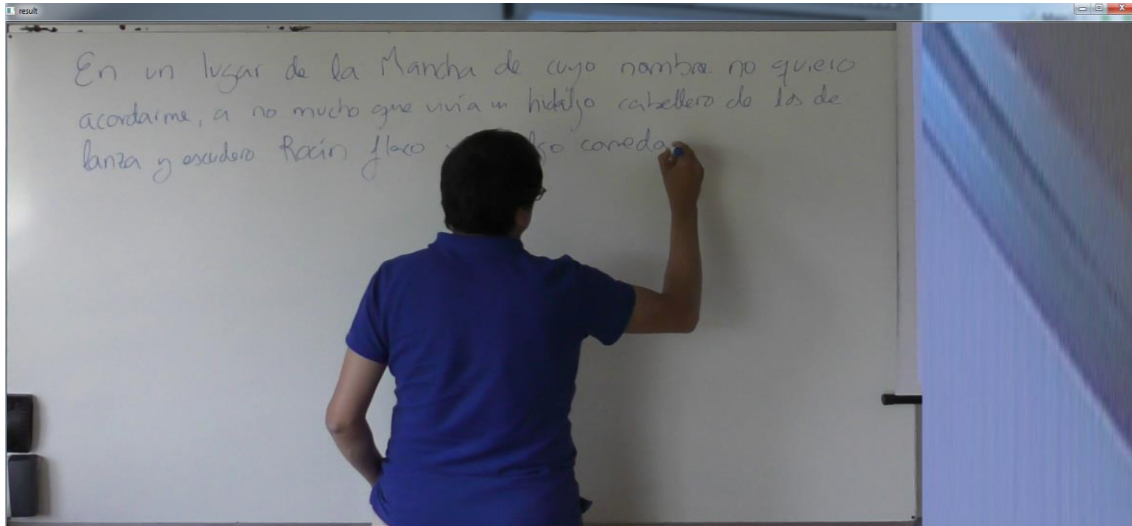


Figura 26, imagen de la pizarra y el profesor unidos. Usando Kernel igual a tres

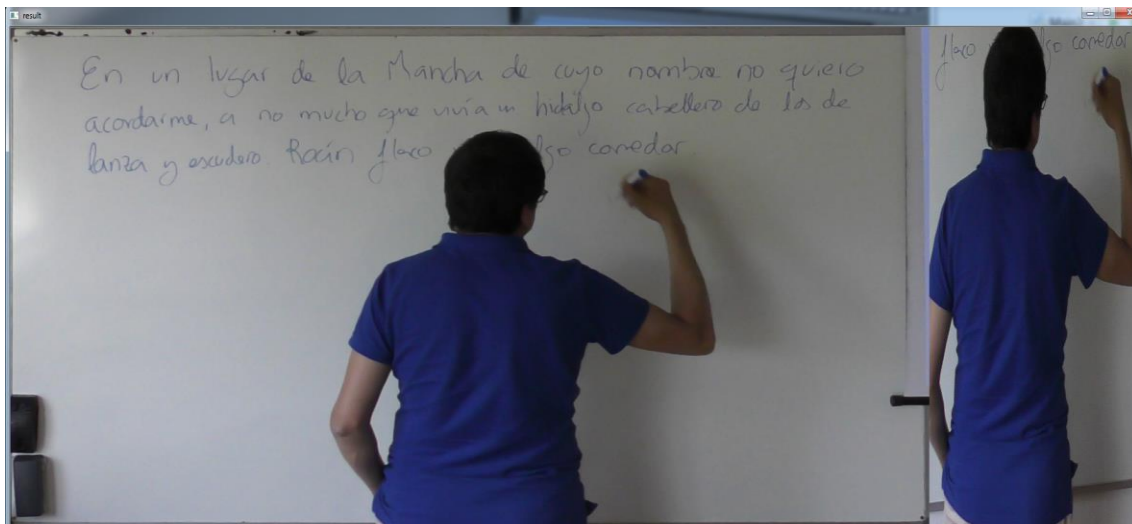


Figura 27, imagen de la pizarra y el profesor unidos. Usando un Kernel igual a treinta y uno

- Umbralizado:

Los límites del umbral son muy delicados y dependen mucho del tipo de imagen. Por este motivo, entre un vídeo y otro, la diferencia es el tamaño del Kernel que hace referencia al gaussianBlur y el límite del umbral. Dependiendo del límite del umbral, se crean más o menos posibles contornos y por tanto más o menos falsos positivos. Al subir el límite de 80 a 100 se encuentran falsos positivos ya que el umbral no filtra lo suficiente (ver Figura 28 y Figura 29).



Figura 28, imagen umbralizada errónea

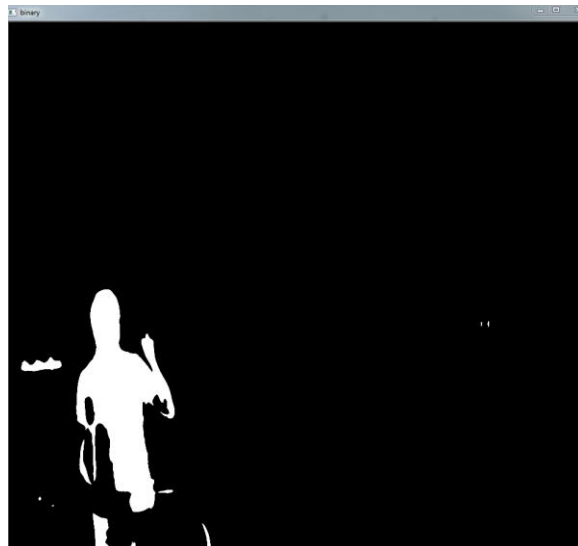


Figura 29, imagen umbralizada correcta

- Obtención de contornos.

Sin duda otro reto ha sido la elección del área mínima, de cara a no obtener falsos positivos mediante la obtención de contornos con la imagen umbralizada. En este caso se ha seleccionado un valor de 10.000 que permitía identificar fácilmente al profesor.

Además, como filtro extra, se ha procedido a eliminar falsos positivos dependiendo de la posición del profesor encontrado anteriormente. Para realizar este filtro, se asume una distancia máxima de paso del profesor sobre el eje x y una distancia máxima de salto sobre el eje y. El profesor no puede pasar de una posición x en la imagen a una posición mayor que $x+200$.

Para ilustrar este ejemplo, se adjuntan dos imágenes (ver Figura 30 y Figura 31). Como se puede apreciar en ellas, el vídeo comienza inicializando el ordenador con bastante iluminación como muestra la Figura 30. En este caso, tenemos al profesor identificado

como muestra el contorno que se dibuja sobre él y por tanto el vídeo del profesor será mostrado sin ningún problema. El contorno marcado sobre el ordenador que aparece a la izquierda del vídeo no supondría un problema ya que posteriormente será marcado como falso profesor y eliminado puesto que el área es más pequeña que el área deseada.

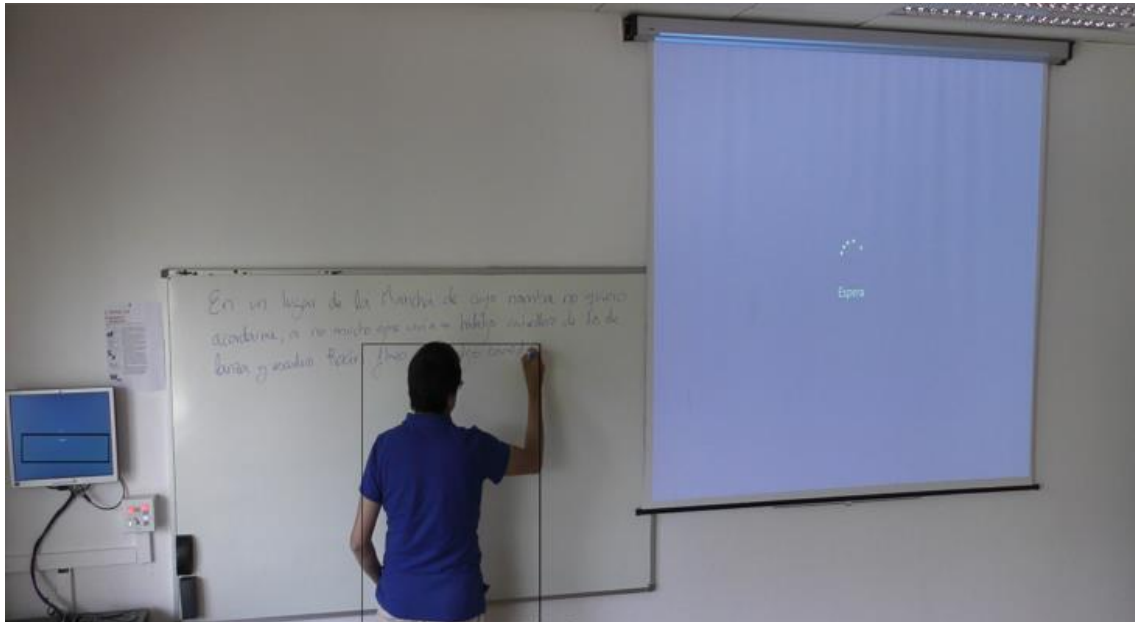


Figura 30, imagen de contornos encontrados en la búsqueda del profesor

Sin embargo, una vez que el ordenador ha inicializado (ver Figura 31), la iluminación cambia considerablemente y el software encontraba contornos posibles positivos que no lo eran. Algunos de estos contornos tienen un área igual o superior al área del profesor por lo que inicialmente podrían causar problemas. Para eliminar este problema de falsos positivos encontrados debido al cambio de iluminación, el software mantiene la posición anterior del profesor. Al recordar esta posición y estableciendo un paso máximo al que el profesor podría andar o saltar a lo largo de la clase, se ha aplicado otro filtro que descarta estos contornos quedándose con el contorno del profesor verdadero.



Figura 31, imagen de contornos encontrados en la búsqueda del profesor sin aplicar el filtro

Gracias a este último filtro pasaríamos de tener un falso positivo como podemos ver en la Figura 32, a un positivo verdadero como podemos ver en la Figura 33, ya que el contorno encontrado previamente no es encontrado posteriormente (ver Figura 34).

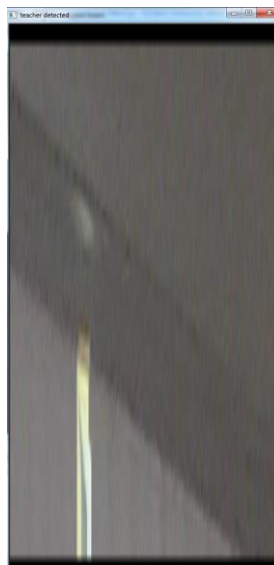


Figura 32, falso positivo del profesor

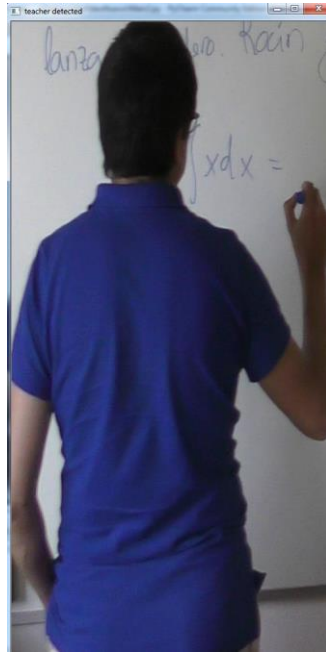


Figura 33, positivo verdadero del profesor

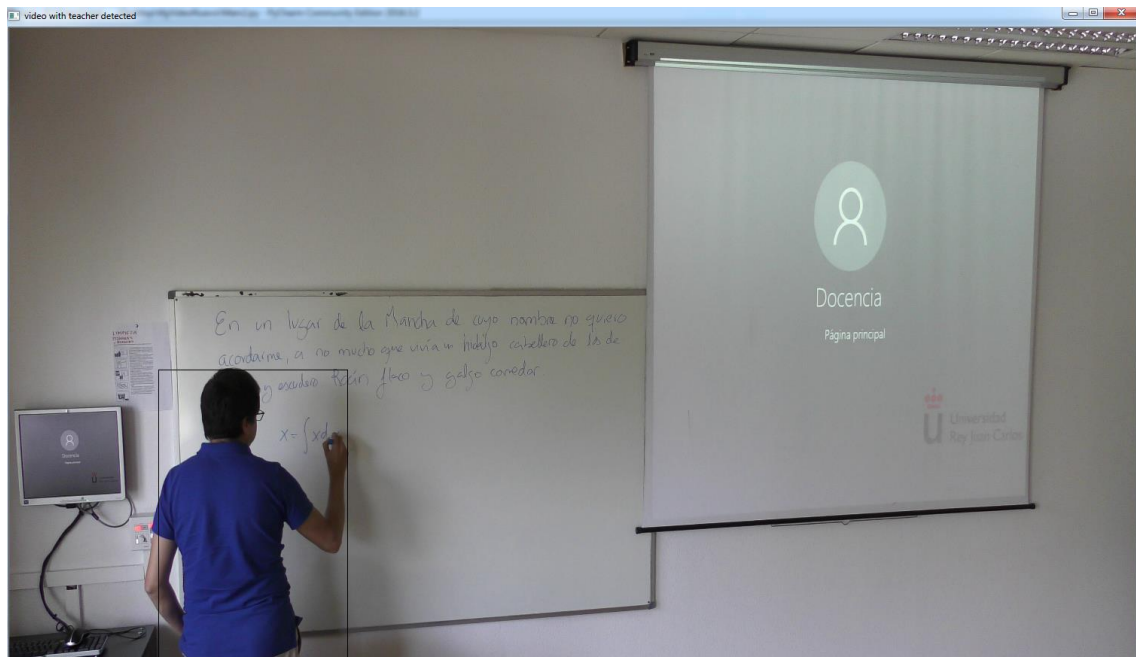


Figura 34, imagen de contorno encontrado en la búsqueda del profesor tras aplicar el filtro

5. Métricas

A la hora de realizar este trabajo se destacan dos tipos de métricas: tiempo empleado y líneas de código.

5.1 Tiempo empleado

La mayor parte del tiempo empleado en este TFG se corresponde a la realización del diseño y pruebas como se puede ver en la Figura 35.

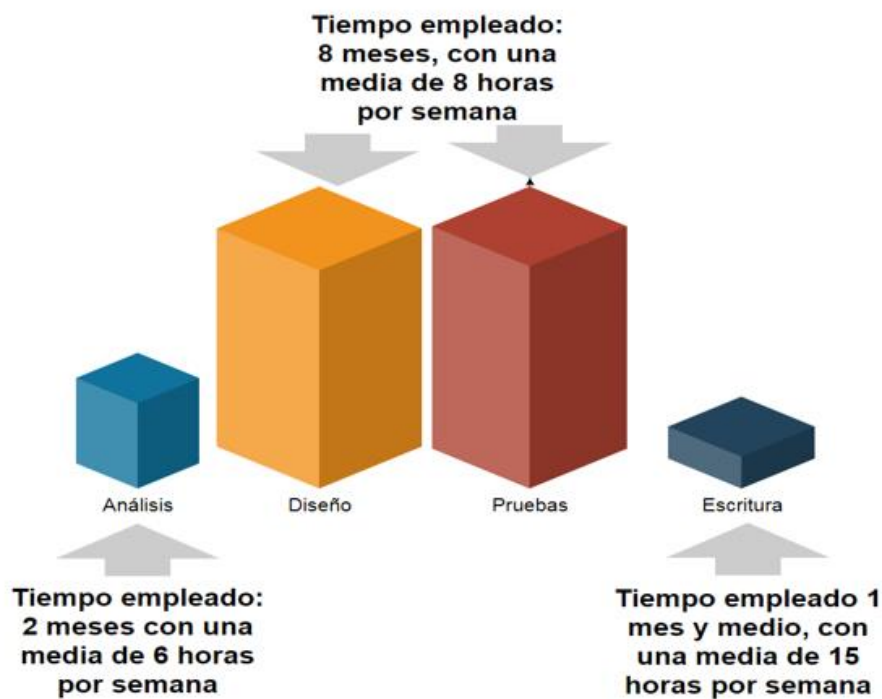


Figura 35, tiempo empleado

La mayor parte del tiempo ha sido invertido en el diseño y las pruebas del mismo. El análisis del problema, aunque ha sido una iteración difícil ha consumido menos tiempo ya que parte del mismo estaba ya definido por el profesor.

La escritura, aunque ha sido un trabajo tedioso, ha consumido menos tiempo en comparación con las otras partes.

5.2 Métricas del código

En cuanto al código, la métrica se establece con referencia al número de líneas por tarea desarrollada como queda reflejado en la Figura 36.

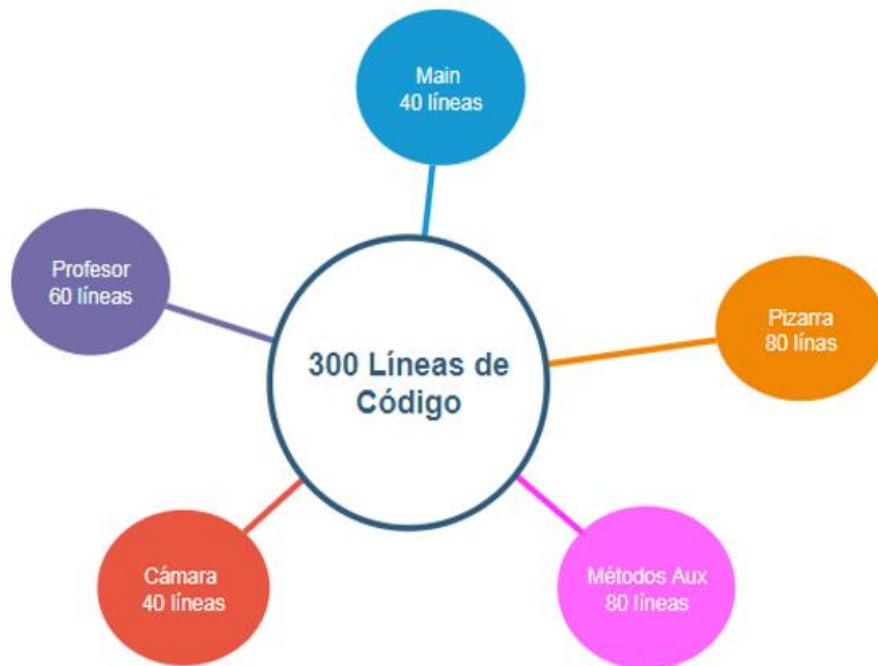


Figura 36, líneas de código

Como muestra la Figura 36, el número de líneas final es de trescientas, las cuales corresponden a las diecisiete funciones que forman las tres clases desarrolladas.

Lo más complejo de este proyecto ha sido usar los métodos necesarios en cada momento para que el software funcionara correctamente. Para poder desarrollar este software se han invertido alrededor de quinientas horas de investigación para el estudio de los métodos y funciones empleadas.

Se debe tener en cuenta que el código desarrollado es de muy alto nivel, pues se apoya en bibliotecas con alta capacidad expresiva como OpenCV y Numpy.

6. Conclusión

Como se ha podido observar a lo largo de la memoria, todos los objetivos planteados han sido alcanzados.

Por un lado se ha podido segmentar la pizarra, corregir su posición y maximizar su tamaño así como mejorar su calidad.

Por otro, se ha logrado segmentar al profesor eliminando todos los errores posibles detectados, dando como resultado final el seguimiento constante del profesor a lo largo de todo el vídeo.

El software desarrollado se ha probado con varios vídeos de distinta resolución. Se considera que el resultado obtenido ha sido satisfactorio en cuanto a los objetivos de segmentación y al requisito de tiempo de ejecución razonable. Si bien es cierto que este último aspecto, el tiempo de ejecución, es mejorable, sobre todo en el caso de los vídeos de 4k.

La Figura 37 y Figura 38 muestran el resultado obtenido sobre un frame de un vídeo de test. En la Figura 37, correspondiente a la segmentación de la pizarra, se puede observar que el profesor aparece sobre ella. No se ha eliminado la presencia del profesor sobre la pizarra ya que sería posible que el profesor escribiera sobre ella y borrarse antes de aparecer en otra parte del aula y esto supondría una pérdida de información para el alumno que atienda esta clase online.

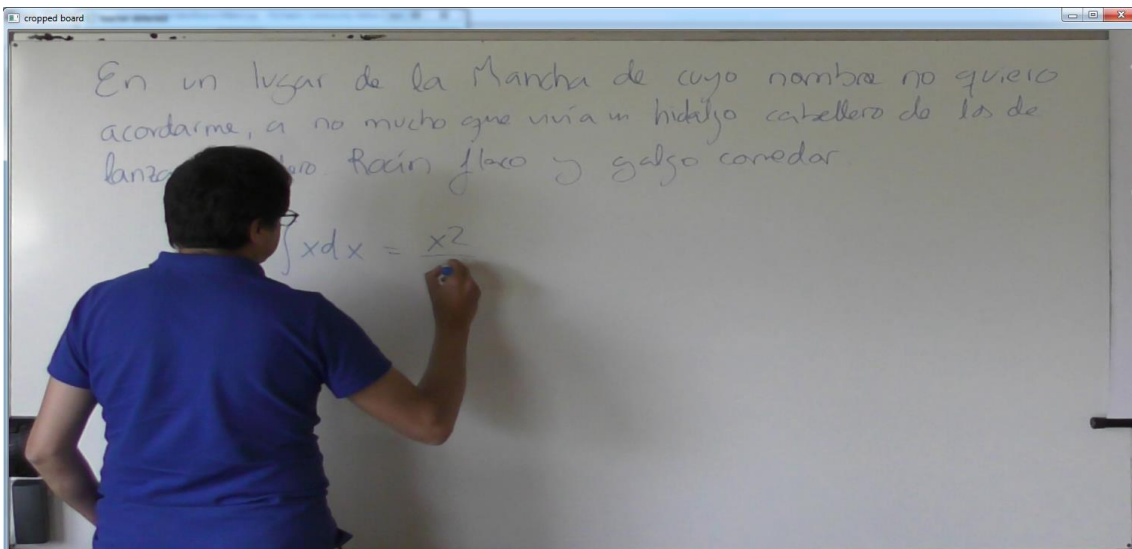


Figura 37, objetivo alcanzado en la segmentación de la pizarra



Figura 38, objetivo alcanzado en la segmentación del profesor

6.1 Trabajos futuros

Como todo software desarrollado, se pueden establecer una serie de mejoras o ampliaciones para el futuro como pueden ser:

- Añadir la configuración inicial del software de una manera más visual a través de una aplicación simple de escritorio en la que se requieran esos parámetros. Esta mejora daría más seguridad al software ya que el técnico de visión no tendría que acceder al código base para modificarlos.
- Extender los vídeos ofertando una mayor inclusividad a través de sistemas de braille o subtítulos. Esta mejora haría que la universidad pudiera ofertar cursos para personas con algún tipo de discapacidad.
- Para trabajos más avanzados se podría utilizar directamente en streaming calculando previamente unos parámetros de configuración.
- Añadir un sistema a través del cual, los alumnos desde su casa tengan opción a hacer una señal para pedir permiso para dirigir una duda al profesor y que este pueda, en tiempo real, recibir la petición y poder exponerla a todos para aclarar una duda o una sugerencia.

7. Bibliografía

[1] Muralidhar Rangaswamy Fred Harris, Digital Signal Processing, pág 414-432, 2006 [En línea] [Consultado el 11/02/2020] Disponible en: <https://www.microsoft.com/en-us/research/uploads/prod/2016/11/Digital-Signal-Processing.pdf>

[2] Erwin M.Bakker, Thomas S.Huang, Image and Video Retrieval, pág 662-365, 2003 Second International Conference on Image Processing [en línea][Consultado el 11/02/202] Disponible en: https://books.google.es/books?id=IDttCQAAQBAJ&pg=PA365&lpg=PA365&dq=segmenting+board+and+teacher+in+video+of+a+class&source=bl&ots=U_qWBU5-Tg&sig=ACfU3U1DCn1je46KQCIMrWKY13oP6nkJzQ&hl=es&sa=X&ved=2ahUKEwji4ciThrnpAhWO3eAKHXUUApcQ6AEwGHoECAoQAQ#v=onepage&q=segmenting%20board%20and%20teacher%20in%20video%20of%20a%20class&f=false

[3] Universidad autónoma de Barcelona. [En línea] [Consultado el 11/02/2020] Disponible en: <https://www.uab.cat/web/estudiar/mooc/-que-es-un-curso-mooc-1345668281247.html>

[4] The MOOC Guide. [En línea] [Consultado el 11/02/2020] Disponible en: <https://sites.google.com/site/themoocguide/3-cck08---the-distributed-course>

[5] Tipos de MOOCs. [En línea] [Consultado el 11/02/2020] Disponible en: <https://sites.google.com/site/losmooc/tipos>

[6] Coursera. Web Oficial [en línea] [Consultado el 11/02/2020] Disponible en: <https://www.coursera.org/>

[7] Miriadax. Web Oficial [en línea] [Consultado el 11/02/2020] Disponible en: <https://miriadax.net>

[8] Adslzone. Web de referencia [en línea] [Consultado el 11/02/2020] Disponible en: <https://www.adslzone.net/reportajes/foto-video/que-es-tecnologia-4k/>

[9] Jetbrains. Web Oficial [en línea] [Consultado el 11/02/2020] Disponible en: <https://www.jetbrains.com/es-es/pycharm/>

[10] Pmis-consulting. Web de clases de Agile online y consultoría [en línea] [Consultado el 11/02/2020] Disponible en: <https://www.pmis-consulting.com/why-agile-will-never-be-a-project-management-framework/>

[11] Managedagile. Web de clases de Agile online [en línea] [Consultado el 11/02/2020] Disponible en: <https://managedagile.com/what-are-the-advantages-and-disadvantages-of-agile-scrum/>

[12] Python. Web oficial [en línea] [Consultado el 11/02/2020] Disponible en: <https://www.python.org/>

- [13] Makeitreal. Web cursos online [en línea] [Consultado el 11/02/2020] Disponible en: <https://blog.makeitreal.camp/lenguajes-compilados-e-interpretados/>
- [14] Pypl. Análisis de popularidad de lenguajes de programación [en línea] [Consultado el 11/02/2020] Disponible en: <http://pypl.github.io/PYPL.html>
- [15] mtp. Artículo Python [en línea] [Consultado el 11/02/2020] Disponible en: <https://www.mtp.es/python-un-lenguaje-de-programacion-para-favorecer-la-legibilidad-del-codigo>
- [16] Computerhoy. Web de tecnología [en línea] [Consultado el 11/02/2020] Disponible en: <http://www.google.es/amp/s/computerhoy.com/noticias/tecnologia/python-adelantara-c-java-como-lenguaje-programacion-cuestion-anos-437455%3famp>
- [17] OpenCV. Web Oficial [en línea] [Consultado el 11/02/2020] Disponible en: <https://opencv.org/about/>
- [18] Numpy. Web Oficial [en línea] [Consultado el 11/02/2020] Disponible en: <https://numpy.org/>
- [19] hackr. Blog tecnología [en línea] [Consultado el 11/02/2020] Disponible en: <https://hackr.io/blog/top-data-science-python-libraries>
- [20] Wikipedia. Web Oficial [en línea] [Consultado el 11/02/2020] Disponible en: https://es.wikipedia.org/wiki/Desenfoque_gaussiano
- [21] Geometric_transformations. Web Oficial [en línea] [Consultado el 11/02/2020] Disponible en: https://docs.opencv.org/2.4/modules/imgproc/doc/geometric_transformations.html
- [22] MAKEITREAL. Lenguajes compilados e interpretados. [En línea][Consultado el 11/02/2020] Disponible en: <https://blog.makeitreal.camp/lenguajes-compilados-e-interpretados/>
- [23] EDN. Practical UML: a Hands-On Introduction for Developers [en línea] [consultado el 11/02/2020] Disponible en: <https://edn.embarcadero.com/article/31863>
- [24] INSTAGRAM. Web oficial con el tag Opencv [en línea] [consultado el 11/02/2020] Disponible en <https://www.instagram.com/explore/tags/opencv/?hl=es>
- [25] SWAROOPCH. A Byte of Python [en línea] [Consultado el 11/02/2020] Disponible en: <https://swaroopch.com/>
- [26] TUTORIALSPPOINT.UML – Activity diagrams [en línea] [Consultado el 11/02/2020] Disponible en: https://www.tutorialspoint.com/uml/uml_activity_diagram.htm
- [27] LIKEGEEKS. Python Image Processing Tutorial (Using OpenCV) [en línea] [Consultado el 11/02/2020] Disponible en: <https://likegeeks.com/python-image-processing/>

[28] Halterman, R. (2019). Fundamentals of Python Programming. Southern Adventist University and Internet Archive.

[29] Wikipedia. Web Oficial [en línea] [Consultado el 11/02/2020] Disponible en: https://en.wikipedia.org/wiki/Deployment_diagram

Anexo I, Configuración de la instalación

Para facilitar el ajuste de este software al técnico de visión que se encargue de ello, se añade este anexo en el que se explica cómo ajustar los parámetros de la pizarra para que el vídeo funcione correctamente.

La pizarra aparecerá en diferentes posiciones dependiendo de la orientación de la cámara que grabe el vídeo de la clase.

Para ajustar estos parámetros se dispone de un archivo .XML llamado CAMERA.XML (ver Figura 39; **Error! No se encuentra el origen de la referencia.**) en la carpeta BoardDoc. Como se puede ver, este archivo contiene la posición de la pizarra en el vídeo (PizarraImagen) y las dimensiones reales de la misma (PizarraReal).

Las dimensiones de la pizarra al ser estándar, no deberán ser modificadas. Sin embargo, las coordenadas de la pizarra en el vídeo son diferentes. Para encontrar estos valores se ha hecho una captura de una imagen del vídeo (ver Figura 40). Con esta imagen se ha procedido a buscar las coordenadas para posteriormente introducirlas en el archivo XML.

```
<xml>
  <PizarraImagen a_x="172" a_y="78" b_x="689" b_y="81" c_x="672" c_y="354" d_x="237" d_y="353"/>
  <PizarraReal width="100" height="50"/>
</xml>
```

Figura 39, archivo XML de configuración de la pizarra

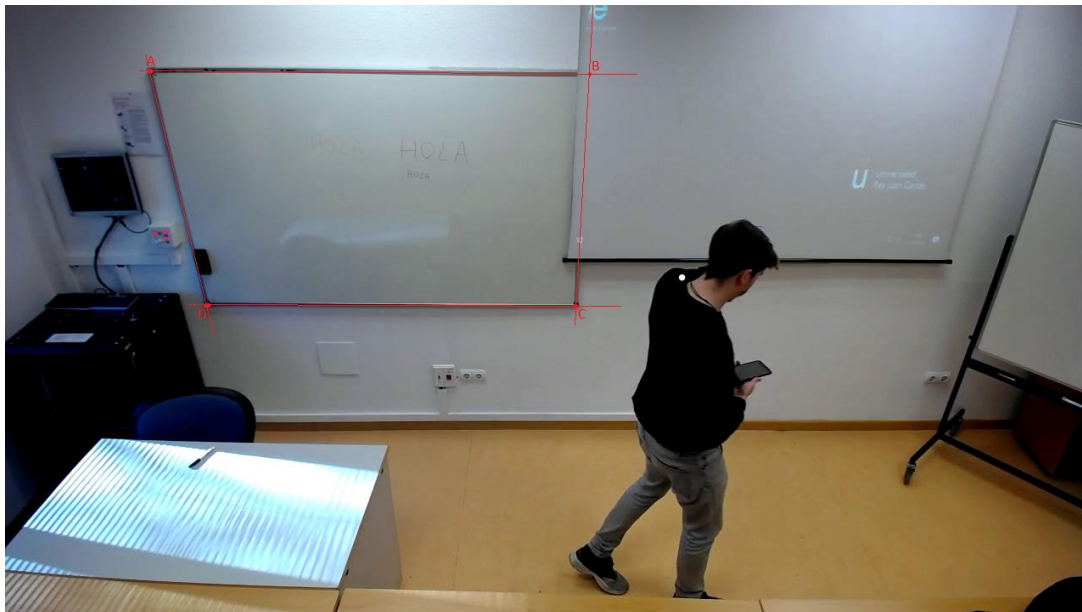


Figura 40, captura de imagen

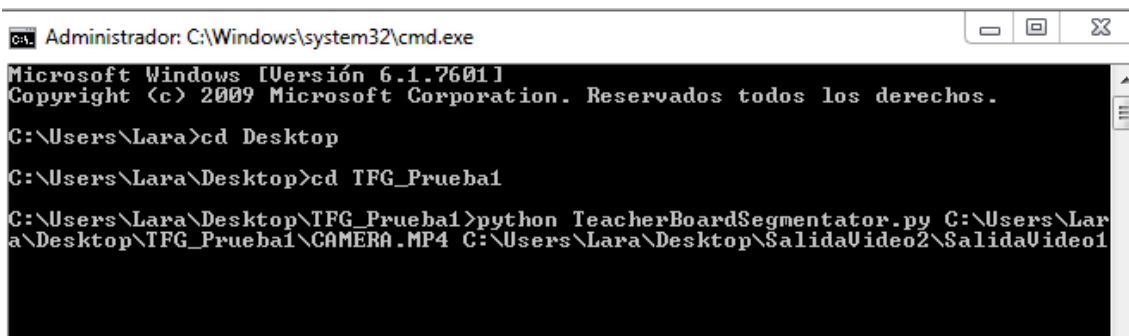
Anexo II, Ejecución de la aplicación

Esta aplicación puede ser ejecutada desde la pantalla de comandos. Para ello será necesaria la siguiente sentencia:

```
python TeacherBoardSegmentator.py <ruta donde se encuentra el vídeo> <directorio final donde se guardarán los vídeos creados>
```

Tras finalizar la ejecución se generarán dos vídeos correspondientes a la pizarra y al profesor segmentado (Board_nombreVídeo y TeacherOut nombreVídeo) generados a partir del vídeo original.

Sirva como ejemplo de ejecución el que se representa a continuación en la Figura 41:



```
Administrador: C:\Windows\system32\cmd.exe
Microsoft Windows [Versión 6.1.7601]
Copyright (c) 2009 Microsoft Corporation. Reservados todos los derechos.

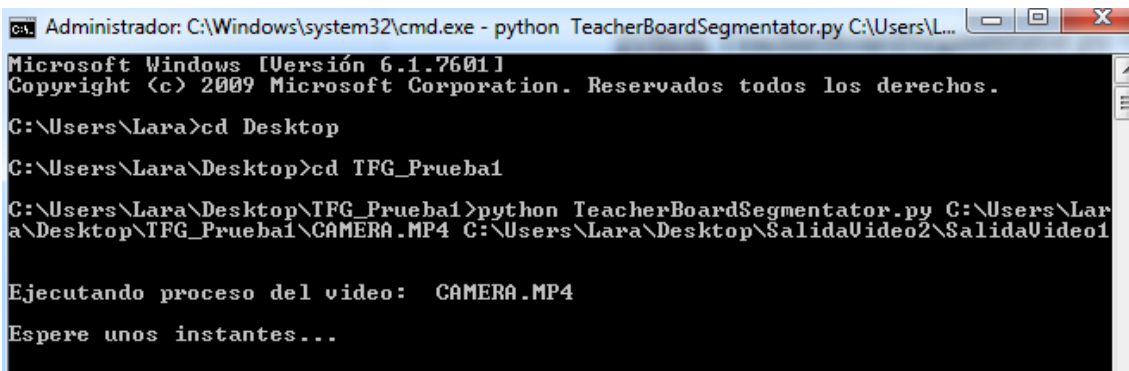
C:\Users\Lara>cd Desktop
C:\Users\Lara\Desktop>cd TFG_Prueba1
C:\Users\Lara\Desktop\TFG_Prueba1>python TeacherBoardSegmentator.py C:\Users\Lara\Desktop\TFG_Prueba1\CAMERA.MP4 C:\Users\Lara\Desktop\SalidaVideo2\SalidaVideo1
```

Figura 41, ejecución de la aplicación desde línea de comandos

Como se puede ver, los pasos son los siguientes:

- Accedemos a la carpeta donde se encuentra alojado nuestra aplicación: TFG_Prueba1
- Ejecutamos la sentencia
python TeacherBoardSegmentator.py
C:\Users\Lara\Desktop\TFG_Prueba1\CAMERA.MP4
C:\Users\Lara\Desktop\SalidaVideo2\SalidaVideo1
- Los resultados serán guardados en la ruta seleccionada, SalidaVideo1

Una vez ejecutada la sentencia, nos aparecerá un aviso para afirmar que el proceso está en ejecución (ver Figura 42).



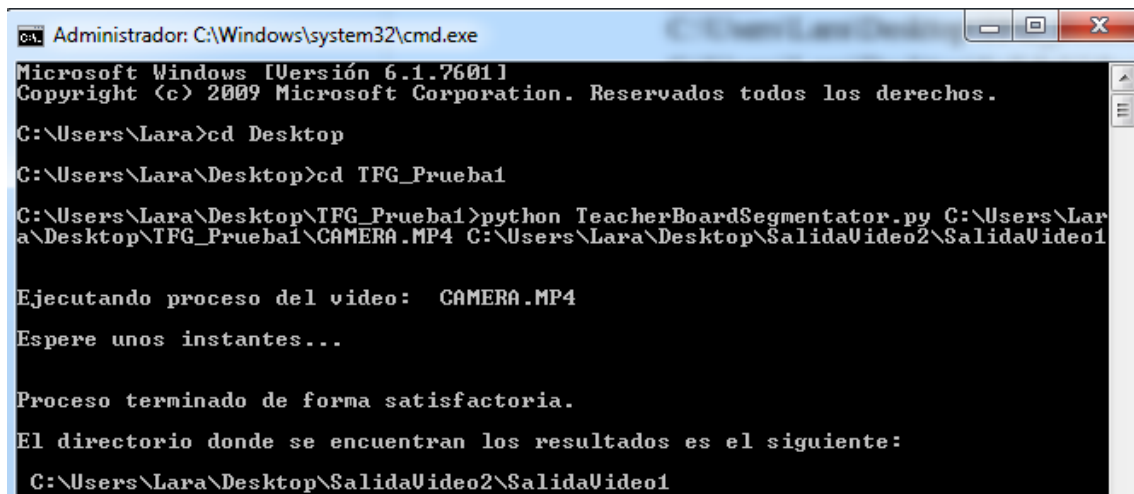
```
Administrador: C:\Windows\system32\cmd.exe - python TeacherBoardSegmentator.py C:\Users\L...
Microsoft Windows [Versión 6.1.7601]
Copyright (c) 2009 Microsoft Corporation. Reservados todos los derechos.

C:\Users\Lara>cd Desktop
C:\Users\Lara\Desktop>cd TFG_Prueba1
C:\Users\Lara\Desktop\TFG_Prueba1>python TeacherBoardSegmentator.py C:\Users\Lara\Desktop\TFG_Prueba1\CAMERA.MP4 C:\Users\Lara\Desktop\SalidaVideo2\SalidaVideo1

Ejecutando proceso del video: CAMERA.MP4
Espere unos instantes...
```

Figura 42, pantalla de comandos tras ejecutar el programa

Cuando el proceso haya terminado, tendremos otro aviso en la pantalla de comandos en el que nos indicará su finalización así como la ruta donde los vídeos generados han sido almacenados (ver Figura 43).



```
C:\Windows\system32\cmd.exe
Microsoft Windows [Versión 6.1.7601]
Copyright (c) 2009 Microsoft Corporation. Reservados todos los derechos.

C:\Users\Lara>cd Desktop
C:\Users\Lara\Desktop>cd TFG_Prueba1
C:\Users\Lara\Desktop\TFG_Prueba1>python TeacherBoardSegmentator.py C:\Users\Lara\Desktop\TFG_Prueba1\CAMERA.MP4 C:\Users\Lara\Desktop\SalidaVideo2\SalidaVideo1

Ejecutando proceso del video: CAMERA.MP4
Espere unos instantes...

Proceso terminado de forma satisfactoria.
El directorio donde se encuentran los resultados es el siguiente:
C:\Users\Lara\Desktop\SalidaVideo2\SalidaVideo1
```

Figura 43, pantalla de comandos una vez terminada la ejecución

Para poder visualizar los resultados bastará con ir a ese directorio y abrirlos (ver Figura 44). Los vídeos generados ya estarían listos para su futuro uso.

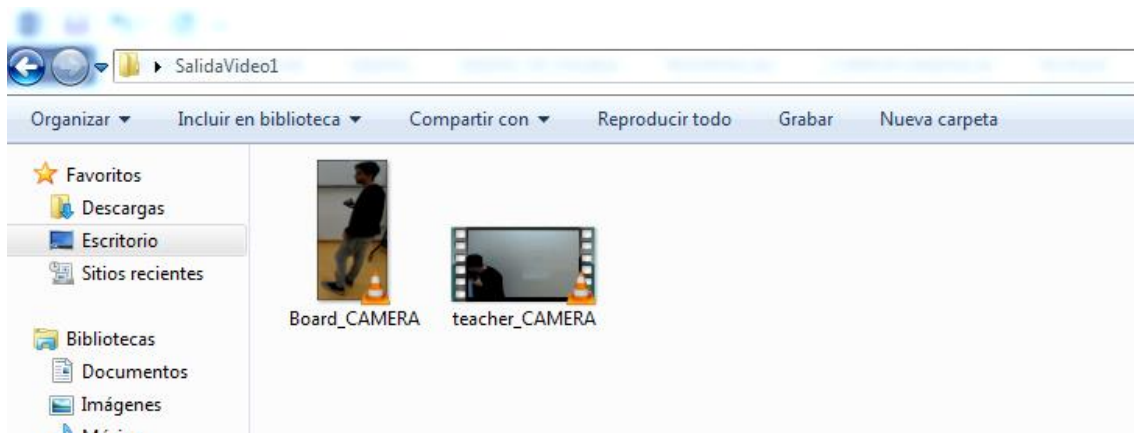


Figura 44, resultados guardados en el directorio deseado